Learning Shortest Paths for Word Graphs

Emmanouil Tzouridis and Ulf Brefeld

Technische Universität Darmstadt,
Hochschulstr. 10, 64289 Darmstadt, Germany
{tzouridis,brefeld}@kma.informatik.tu-darmstadt.de

Abstract. The vast amount of information on the Web drives the need for aggregation and summarisation techniques. We study event extraction as a text summarisation task using redundant sentences which is also known as sentence compression. Given a set of sentences describing the same event, we aim at generating a summarisation that is (i) a single sentence, (ii) simply structured and easily understandable, and (iii) minimal in terms of the number of words/tokens. Existing approaches for sentence compression are often based on finding the shortest path in word graphs that is spanned by related input sentences. These approaches, however, deploy manually crafted heuristics for edge weights and lack theoretical justification. In this paper, we cast sentence compression as a structured prediction problem. Edges of the compression graph are represented by features drawn from adjacent nodes so that corresponding weights are learned by a generalised linear model. Decoding is performed in polynomial time by a generalised shortest path algorithm using loss augmented inference. We report on preliminary results on artificial and real world data.

1 Introduction

Information is ubiquitous. People are consuming information on the go by reading news articles, blogs entries, checking status updates, or planning the next vacations. However, the informed mobility comes at the cost of relevance. Users need to manually identify relevant pieces of information in the overloaded supply and to aggregate these pieces themselves to find an answer to their query. Thus, there is a real need for techniques that assist the user in separating relevant from irrelevant information and aggregating the pieces of information automatically.

In this paper we study the intelligent aggregation of related sentences to quickly serve the information needs of users. Given a collection of sentences dealing with the same real-world event, we aim at generating a single sentence that is (i) a summarisation of the input sentences, (ii) simply structured and easily understandable, and (iii) minimal in terms of the number of words/tokens. The input collection of sentences is represented as a word graph [4], where words are identified with nodes and directed edges connect adjacent words in at least one sentence. The output is a sequence of words fulfilling conditions (i-iii).

In general, learning such mappings between arbitrary structured and interdependent input and output spaces challenges the standard model of learning a

mapping from independently drawn instances to a small set of labels. In order to capture the involved dependencies it is helpful to represent inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$ in a joint feature representation. The learning task is therefore rephrased as finding a function $f : \mathcal{X} \times \mathcal{Y} \to \Re$ such that

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \tag{1}$$

is the desired output for any input $\mathbf{x}$ [2, 12]. The function f is a linear model in a joint space $\Phi(\mathbf{x}, \mathbf{y})$ of input and output variables and the computation of the argmax is performed by an appropriate decoding strategy. Prominent applications of such models are part-of-speech tagging, parsing, or image segmentation.

In this paper, we cast sentence compression as a supervised structured prediction problem to learn a mapping from word graphs to their shortest paths. Edges of the graphs are labeled with costs and the shortest path realises the lowest possible costs from a start to an end node. Thus, we aim at finding a function f, such that

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}). \tag{2}$$

We devise structured perceptrons and support vector machines for learning the shortest path. The latter usually requires the use of two-best decoding algorithms which are expensive in terms of computation time and memory. We proof that an equivalent expression can be obtained by a generalised shortest path algorithm using loss-augmented inference. The latter renders learning much more efficient than using a two-best decoding strategy. Empirically, we compare our approach to the state-of-the-art that uses heuristic edge weights [4] on artificial and real world data and report on preliminary results.

The remainder is structured as follows. Section 2 introduces preliminaries. Our main contribution on learning shortest paths is presented in Section 3 and Section 4 reports on empirical results. Section 5 discusses related work and Section 6 concludes.

2 Preliminaries

2.1 Word Graphs

Word or compression graphs have been studied for instance by [3, 4]. The idea is to build a non-redundant representation for possibly redundant sequences by merging identical observations. From a collection of related sentences $\{s_1, \ldots, s_n\}$, we iteratively construct a word graph by adding sentences one-by-one as follows: We begin with the empty graph and add the first sentence s_1, where every word in the sentence becomes a node and a directed edge connects nodes of adjacent words. After this step, the graph is a sequence representing s_1. Words from the second sentence s_2 are incorporated by differentiating two cases: (i) if the graph already contains the word (e.g., using lowercase representations), the word is

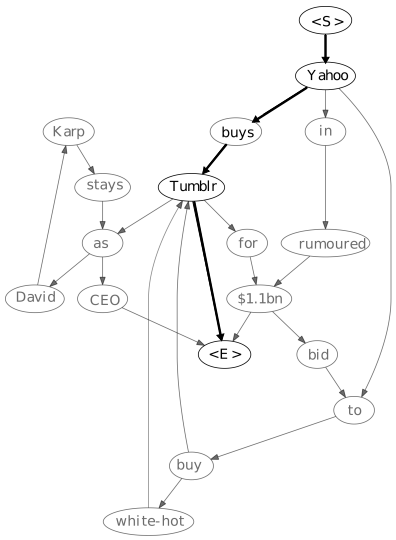

Fig. 1: Word graph constructed from the sentences: "Yahoo in rumoured $1.1bn bid to buy white-hot Tumblr", "Yahoo buys Tumblr as David Karp stays as CEO", "Yahoo to buy Tumblr for $1.1bn". The shortest path is highlighted.

simply mapped to the corresponding node and (ii) otherwise, a new node is instantiated. In both cases, a directed edge is inserted to connect the word to its predecessor from s_2. The procedure continues until all n sentences are incorporated in the graph.

Usually, sentences are augmented by designated auxiliary words indicating the start and the end of the sentence. The sketched procedure merges identical words but preserves the structure of the sentences along the contained paths and the original sentences can often be reconstructed from the compressed representation. There are many different ways to build such graphs, e.g., by excluding punctuations, stop-word removal, or by using part-of-speech information to improve merge operations. Figure 1 shows some related sentences and the corresponding word graph.

2.2 Shortest Path Algorithms

Given a directed weighted graph $\mathbf{x} = (N, E)$, where N is the set of nodes and E the set of edges. As the graph $\mathbf{x}$ defines the sets N and E, we will use $N(\mathbf{x})$ and $E(\mathbf{x})$ in the remainder to denote the set of nodes and edges of graph $\mathbf{x}$, respectively. Every edge $(x_i, x_j) \in E(\mathbf{x})$ is assigned a positive weight given by a cost function $cost : (x_i, x_j) \mapsto \Re^+$. A path $\mathbf{y}$ in the graph $\mathbf{x}$ is a sequence of connected nodes of $\mathbf{x}$ and the cost of such a path is given by the sum of the edge costs for every edge that is on the path. Given the graph $\mathbf{x}$, a start node x_s and

an end node x_e, the shortest path problem is finding the path in $\mathbf{x}$ from x_s to x_e with the lowest costs,

$$\begin{aligned} \underset{\mathbf{y}}{\operatorname{argmin}} \quad & \sum_{(x_i,x_y)\in N(\mathbf{x})} y_{ij} cost(x_i, x_{i+1}) \\ \text{s.t.} \quad & \mathbf{y} \in path(x_s, x_e). \end{aligned}$$

There exist many algorithms for computing shortest paths efficiently [6–8]. Usually, these methods are based on relaxation integer programming, where an approximation of the exact quantity is iteratively updated until it converges to the correct solution. Such algorithms converge in polynomial time if there are no negative cycles inside the graph, otherwise the problem is NP-hard. A prominent algorithm for computing the k-th shortest paths is Yen's algorithm [5]. Intuitively, the approach recursively computes the second best solution by considering deviations from the shortest path, the third best solution from the previous two solutions, and so on. Figure 1 visualises the shortest path for the displayed compression graph.

3 Learning the Shortest Path

3.1 Representation

To learn the shortest path, we need to draw features from adjacent nodes in the word graph to learn the score of the connecting edge. Let x_i and x_j be connected nodes of the compression graph $\mathbf{x}$, that is $x_i, x_j \in N(\mathbf{x})$ and $(x_i, x_j) \in E(\mathbf{x})$. We represent the edge between x_i and x_j by a feature vector $\phi(x_i, x_j)$ that captures characteristic traits of the connected nodes, such as indicator functions detailing whether x_i and x_j are part of the same named entity or part-of-speech transitions.

A path in the graph is represented as an $n \times n$ binary matrix $\mathbf{y}$ with $n = |N(\mathbf{x})|$ and elements $\{\mathbf{y}_{ij}\}$ given by $y_{ij} = [[(x_i, x_j) \in path]]$ where $[[z]]$ is the indicator function returning one if z is true and zero otherwise. The cost of using the edge (x_i, x_j) in a path is given by a linear combination of those features parameterised by $\mathbf{w}$,

$$cost(x_i, x_j) = \mathbf{w}^\top \phi(x_i, x_j).$$

Replacing the constant costs by the parameterised ones, we arrive at the following objective function (ignoring the constraints for a moment) that can be rewritten as a generalised linear model.

$$\sum_{(x_i,x_j)\in E(\mathbf{x})} y_{ij}\, \mathbf{w}^\top \phi(x_i, x_j) = \mathbf{w}^\top \underbrace{\left(\sum_{(x_i,x_j)\in E(\mathbf{x})} y_{ij} \phi(x_i, x_j) \right)}_{=\Phi(\mathbf{x},\mathbf{y})} = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}, \mathbf{y})$$

Given a word graph $\mathbf{x}$, the shortest path $\hat{\mathbf{y}}$ for a fixed parameter vector $\mathbf{w}$ can now be computed by

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}),$$

where f is exactly the objective of the shortest path algorithm and the argmin consequently computed by an appropriate solver, such as Yen's algorithm [5].

3.2 Problem Setting

In our setting, word graphs $\mathbf{x} \in \mathcal{X}$ and the best summarising sentence $\mathbf{y} \in \mathcal{Y}$ are represented jointly by a feature map $\Phi(\mathbf{x}, \mathbf{y})$ that allows to capture multiple-way dependencies between inputs and outputs. We apply a generalised linear model $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y})$ to decode the shortest path

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}).$$

The quality of f is measured by the Hamming loss

$$\Delta_H(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_{(x_i, x_j) \in E(\mathbf{x})} [[y_{ij} \neq \hat{y}_{ij}]]$$

that details the differences between the true $\mathbf{y}$ and the prediction $\hat{\mathbf{y}}$, where $[[\cdot]]$ is again the indicator function from Section 3.1. Thus, the generalisation error is given by

$$R[f] = \int_{\mathcal{X} \times \mathcal{Y}} \Delta_H \left(\mathbf{y}, \operatorname*{argmin}_{\bar{\mathbf{y}}} f(\mathbf{x}, \bar{\mathbf{y}}) \right) dP(\mathbf{x}, \mathbf{y})$$

and approximated by its empirical counterpart

$$\hat{R}[f] = \sum_{i=1}^{m} \Delta_H \left(\mathbf{y}, \operatorname*{argmin}_{\bar{\mathbf{y}}} f(\mathbf{x}, \bar{\mathbf{y}}) \right) \tag{3}$$

on a finite m-sample of pairs $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ...(\mathbf{x}_m, \mathbf{y}_m)$ where $\mathbf{x}_i$ is a word graph and $\mathbf{y}_i$ its shortest path (i.e., the best summarising sentence). Minimising the empirical risk in Equation (3) directly leads to an optimisation problem that is not well-posed as there generally exist many equally well solutions that realise an empirical loss of zero. We thus consider also the minimisation of the regularised empirical risk

$$\hat{Q}[f] = \Omega(f) + \sum_{i=1}^{m} \Delta_H \left(\mathbf{y}, \operatorname*{argmin}_{\bar{\mathbf{y}}} f(\mathbf{x}, \bar{\mathbf{y}}) \right) \tag{4}$$

where $\Omega(f)$ places a prior on f, e.g., to enforce smooth solutions. In the remainder we focus on $\Omega(f) = \|\mathbf{w}\|^2$.

3.3 Perceptron-based Learning

Structured Perceptrons [13, 14] directly minimise the empirical loss in Equation (3). The training set is processed iteratively, where in the i-th iteration the actual prediction $\hat{\mathbf{y}} = \operatorname{argmin}_{\mathbf{y}} f(\mathbf{x}_i, \mathbf{y})$ is compared with the true output $\mathbf{y}_i$. If $\Delta_H(\mathbf{y}_i, \hat{\mathbf{y}}) = 0$ the algorithm proceeds with the next instance. However, if $\Delta_H(\mathbf{y}_i, \hat{\mathbf{y}}) \neq 0$ an erroneous prediction is made and the parameters need to be adjusted accordingly. In case of learning shortest paths, we aim at assigning a smaller function value to the true path than to all alternative paths. If this holds for all training examples we have

$$\forall \quad \bar{\mathbf{y}} \neq \mathbf{y}_i : \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) > 0. \tag{5}$$

In case one of these constraints is violated, the parameter vector is updated according to

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \hat{\mathbf{y}}) - \Phi(\mathbf{x}_i, \mathbf{y}_i),$$

where $\hat{\mathbf{y}}$ denotes the erroneously decoded path (the false prediction). One can show that the perceptron converges if an optimal solution realising $\hat{R}[f] = 0$ exists. [13, 14]

3.4 Large-margin Approach

For support vector-based learning, we extend the constraints in Equation (5) by a term that induces a margin between the true path $\mathbf{y}_i$ and all alternative paths. A common technique is called margin-rescaling and implies to scale the margin with the actual loss that is induced by decoding $\bar{\mathbf{y}}$ instead of $\mathbf{y}_i$. Thus, rescaling the margin by the loss implements the intuition that the confidence of rejecting a mistaken output is proportional to its error. In the context of learning shortest paths, margin-rescaling gives us the following constraint

$$\forall \bar{\mathbf{y}} \neq \mathbf{y}_i : \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) > \Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) - \xi_i, \tag{6}$$

where $\xi_i \geq 0$ is a slack-variable that allows pointwise relaxations of the margin. Solving the equation for ξ_i shows that margin rescaling also effects the hinge loss that now augments the structural loss Δ_H,

$$\ell_{\Delta_H}(\mathbf{x}, \mathbf{y}, f) = \max \left[0, \min_{\bar{\mathbf{y}}} \left[\Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) \right] \right].$$

The effective hinge loss upper bounds the structural loss Δ_H for every pair $(\mathbf{x}_i, \mathbf{y}_i)$ and therefore also

$$\sum_{i=1}^{m} \ell_{\Delta_H}(\mathbf{x}_i, \mathbf{y}_i, f) \geq \sum_{i=1}^{m} \Delta_H(\mathbf{y}_i, \operatorname*{argmin}_{\bar{\mathbf{y}}} f(\mathbf{x}_i, \bar{\mathbf{y}}))$$

holds. Instead of minimising the empirical risk in Equation (3), structural support vector machines [2] aim to minimise its regularised counterpart in Equation

(4). A maximum-margin approach to learning shortest paths therefore leads to the following optimisation problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \quad & \|\mathbf{w}\|^2 + \sum_{i=1}^{m} \xi_i \\ s.t. \quad & \forall i \forall \bar{\mathbf{y}} \neq \mathbf{y}_i : \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) > \Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) - \xi_i \\ & \forall i : \xi_i \geq 0 \end{aligned}$$

The above optimisation problem can be solved by cutting plane methods (e.g., [2]). The idea behind cutting planes is to instantiate only a minimal subset of the exponentially many constrains. That is, for the i-th training example, we decode the shortest path $\hat{\mathbf{y}}$ given our current model and consider two cases: (i) The case $\hat{\mathbf{y}} \neq \mathbf{y}_i$ the prediction is erroneous and $\hat{\mathbf{y}}$ is called the most strongly violated constraint as it realises the smallest function value, i.e., $f(\mathbf{x}_i, \hat{\mathbf{y}}) < f(\mathbf{x}_i, \mathbf{y})$ for all $\mathbf{y} \neq \hat{\mathbf{y}}$. Consequentially, the respective constraint of the above optimisation problem is instantiated and influences the subsequent iterations. (ii) If instead the prediction is correct, that is $\hat{\mathbf{y}} = \mathbf{y}_i$, we need to verify that the second best prediction $\hat{\mathbf{y}}^{(2)}$ fulfils the margin constraint. If so, we proceed with the next training example, otherwise we instantiate the corresponding constraint, analogously to case (i). Luckily, we do not need to rely on an expensive two-best shortest path algorithm but can compute the most strongly violated constraint directly via the cost function

$$Q(\bar{\mathbf{y}}) = \Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) \tag{7}$$

that has to be maximised wrt $\mathbf{y}$. The following proposition shows that we can equivalently solve a shortest path problem for finding the maximiser of Q.

Proposition 1 (Loss augmented inference for shortest path problems). *The maximum $\mathbf{y}^*$ of Q in Equation (7) can be equivalently computed by minimising a shortest path problem with cost$(x_i, x_j) = y_{ij} + \mathbf{w}^\top \phi(x_i, x_j)$.*

Proof. In the proof, we treat paths $\mathbf{y}$ as graphs and write $N(\mathbf{y})$ for the set of nodes on the path and $E(\mathbf{y})$ to denote the set of edges that lie on the path. If, for instance, an element of the binary adjacency matrix representing path $\mathbf{y}$ equals one, e.g., $y_{ij} = 1$, we write $y_i, y_j \in N(\mathbf{y})$ and $(y_i, y_j) \in E(\mathbf{y})$. First, note that the Hamming loss can be rewritten as

$$\Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) = \sum_{(y_i, y_j) \in E(\mathbf{y})} (1 - y_{ij} \bar{y}_{ij}) \,. \tag{8}$$

Using Equation (8), we have

$$
\begin{aligned}
\hat{\mathbf{y}} &= \operatorname*{argmax}_{\bar{\mathbf{y}}} \Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \\
&= \operatorname*{argmax}_{\bar{\mathbf{y}}} \Delta_H(\mathbf{y}_i, \bar{\mathbf{y}}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \\
&= \operatorname*{argmax}_{\bar{\mathbf{y}}} \sum_{(y_i,y_j)\in E(\mathbf{y})} (1 - y_{ij}\bar{y}_{ij}) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \\
&= \operatorname*{argmax}_{\bar{\mathbf{y}}} - \sum_{(y_i,y_j)\in E(\mathbf{y})} y_{ij}\bar{y}_{ij} - \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \\
&= \operatorname*{argmin}_{\bar{\mathbf{y}}} \sum_{(y_i,y_j)\in E(\mathbf{y})} y_{ij}\bar{y}_{ij} + \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \\
&= \operatorname*{argmin}_{\bar{\mathbf{y}}} \sum_{(y_i,y_j)\in E(\mathbf{y})} y_{ij}\bar{y}_{ij} + \mathbf{w}^\top \left(\sum_{(x_i,x_j)\in E(\mathbf{x})} \bar{y}_{ij}\phi(x_i, x_j) \right) \\
&= \operatorname*{argmin}_{\bar{\mathbf{y}}} \sum_{(x_i,x_j)\in E(\mathbf{x})} y_{ij}\bar{y}_{ij} + \mathbf{w}^\top \left(\sum_{(x_i,x_j)\in E(\mathbf{x})} \bar{y}_{ij}\phi(x_i, x_j) \right) \\
&= \operatorname*{argmin}_{\bar{\mathbf{y}}} \sum_{(x_i,x_j)\in E(\mathbf{x})} (y_{ij} + \mathbf{w}^\top \phi(x_i, x_j))\bar{y}_{ij}
\end{aligned}
$$

The output $\hat{\mathbf{y}}$ is the shortest path with costs given by $y_{ij} + \mathbf{w}^\top \phi(x_i, x_j)$. □

Given a parameter vector $\mathbf{w}$ and start and end nodes x_s and x_t, respectively, the optimisation of Q can be performed with the following linear program.

$$
\begin{aligned}
\min_{\bar{\mathbf{y}}} \; & \sum_{ij} \left(y_{ij} + \mathbf{w}^\top \phi(x_i, x_j) \right) \bar{y}_{ij} \\
s.t. \quad & \forall k \in N(\mathbf{x})/\{s,t\} : \; \sum_j \bar{y}_{kj} - \sum_i \bar{y}_{ik} \leq 0 \\
& \forall k \in N(\mathbf{x})/\{s,t\} : - \sum_j \bar{y}_{kj} + \sum_i \bar{y}_{ik} \leq 0 \\
& \sum_j \bar{y}_{sj} - \sum_i \bar{y}_{i,s} \leq 1 \quad \wedge \quad - \sum_j \bar{y}_{sj} + \sum_i \bar{y}_{is} \leq -1 \\
& \sum_i \bar{y}_{it} - \sum_j \bar{y}_{tj} \leq 1 \quad \wedge \quad - \sum_i \bar{y}_{it} + \sum_j \bar{y}_{tj} \leq -1 \\
& \forall (i,j) : \; y_{ij} \leq x_{(i,j)} \quad \wedge \quad \forall (i,j) : \mathbf{y}_{ij} \in \{0,1\}
\end{aligned}
$$

The first two constraints guarantee that every inner node of the path must have as many incoming as outgoing edges, the third line of constraints guarantees the path to start in x_s and, analogously, the forth line ensures that it terminates in x_t. The last line of constraints forces the edges of the path $\bar{\mathbf{y}}$ to move along existing paths of $\mathbf{x}$.

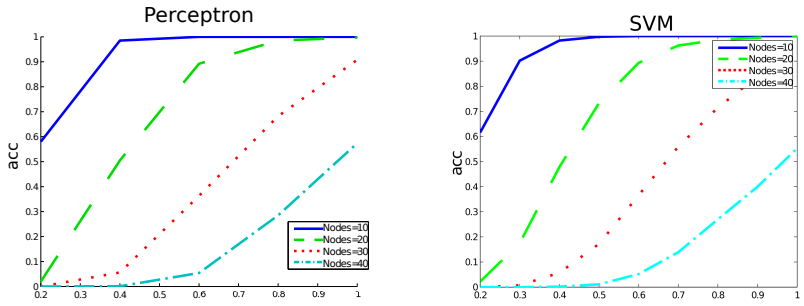

Fig. 2: Performance on artificial data for perceptrons (left) and SVMs (right).

4 Empirical Results

4.1 Artificial Data

To showcase the effectivity of our approach, we generate artificial data as follows. We sample random graphs with $|N| \in \{10, 20, 30, 40\}$ nodes. For every node in a graph, we sample the number of outgoing edges uniformly in the interval $[\frac{|N|}{2}, |N|]$. For every outgoing edge, a receiving node is sampled uniformly from the remaining $|N|-1$ nodes. The optimal path is annotated as follows. We draw the length of the path uniformly in the interval $[\frac{|N|}{2}, |N|]$ and randomly select the respective number of nodes from N. This gives us a sequence of nodes that we define as the shortest path. In case two adjacent nodes are not connected by an edge, we discard the nodes and draw again.

To ensure that the optimal path is actually the one with lowest costs, edge features are sampled from a one-dimensional Gaussian mixture distribution, where the generating component is chosen according to whether the respective edge lies on the the shortest path or not. That is, we introduce two Gaussian components $G_{0,1}$, so that costs for edges lying on the shortest path are drawn from $G_0(\mu_1, \sigma_1^2)$ while costs for all other edges are sampled from $G_1(\mu_2, \sigma_2^2)$.

The difficulty of the experimental setup is controlled by a parameter α that measures the distance of the two means, i.e., $\alpha = |\mu_1 - \mu_2|$. We initialise the components $G_{0,1}$ by drawing one of them (say q) according to a coin flip and sample the corresponding mean from a normal distribution $\mu_q \sim G_q(-\frac{\alpha}{2}, 0.1)$. $G_{\bar{q}}$ is then initialised with $\mu_{\bar{q}} \sim G_{\bar{q}}(\frac{\alpha}{2}, 0.1)$. We use $\sigma_1 = \sigma_2 = 0.01$. We report on averages over 100 repetitions.

The results for perceptrons and SVMs are shown in Figure 2. The distance α is depicted on the x-axis. The y-axis shows the top-one accuracy. The performance of both algorithms highly depends on the distance of the cost-generating components and the size of the graph. The fewer nodes and the larger the distance α, the more accurate is the prediction. Both algorithms perform similarly.

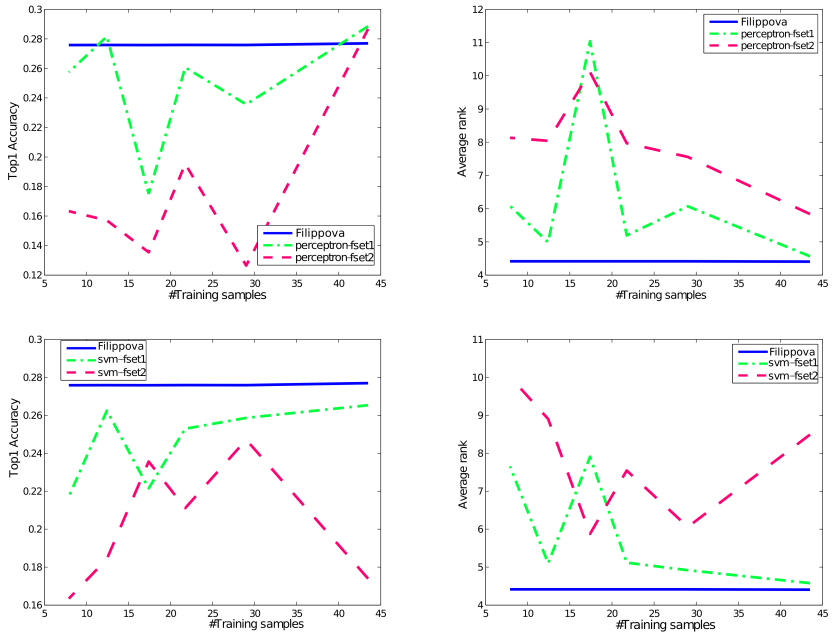

Fig. 3: Performance on news headlines. Accuracies (left column) and average ranks (right column) are evaluated for perceptrons (top row) and SVMs (bottom row)

4.2 News Headlines

The real-world data originates from news articles which have been crawled from several web sites on different days. We use categories *Technology, Sports, Business* and *General* and focus only on the headlines. Related sets of news headlines are manually identified and grouped together. We discard collections with less than 5 headlines and build word graphs for the remaining sets following the procedure described in Section 2.1 where we remove stop words. Ground truth is annotated manually by selecting the best sentence among the 20 shortest paths computed by Yen's algorithm [5] using frequencies as edge weights, this annotation has been done by one of the authors. This process leaves us with 87 training examples.

We aim to learn the costs for the edges that give rise to the optimal compression of the training graphs. We devise three different sets of features. The first feature representation consists of only two features that are inspired by the heuristic in [4]. For an edge (x_i, x_j), we use

$$\phi_1(x_i, x_j) = \left(\frac{\#(x_1)}{\#(x_1, x_2)}, \frac{\#(x_2)}{\#(x_1, x_2)} \right)^\top,$$

Table 1: Leave-one-out results for news headlines

	avg. acc.	avg. rank
Filipova	0.277	4.378
SVM-fset3	0.252	6.942

where # denotes the frequency of nodes and edges, respectively. The second feature representation extends the previous one by Wordnet similarity $sim_W(x_1, x_2)$ of the nodes and frequencies $\#_L$ taken from the Leipzig corpus, to incorporate the notion of global relatedness,

$$\phi_2(x_i, x_j) = \left(\frac{\#(x_1)}{\#(x_1, x_2)}, \frac{\#(x_2)}{\#(x_1, x_2)}, sim_W(x_1, x_2), \frac{\#_L(x_1)}{\#_L(x_1, x_2)}, \frac{\#_L(x_2)}{\#_L(x_1, x_2)} \right)^\top .$$

Finally, we use a third feature representation which is again inspired by [4]. Instead of precomputing the surrogates, we simply input the ingredients to have the algorithm pick the best combination,

$$\phi_3(x_i, x_j) = \left(\#(x_1), \#(x_2), \#(x_1, x_2), \log\left(\#(x_1)\right), \log\left(\#(x_2)\right), \log\left(\#(x_1, x_2)\right)\right)^\top .$$

We compare our algorithms with the unsupervised approach by Filippova [4]. In that work, the author extracts the shortest path from the word-graph that uses $\frac{\#(x_1)+\#(x_2)}{\#(x_1,x_2)}$ as edge weights.

Figure 3 shows average accuracy (left column) and average rank (right column) for perceptrons (top row) and SVMs (bottom row) for different training set sizes, depicted on the x-axis. Every curve is the result of a cross-validation that uses all available data. Thus, the rightmost points are generated by a 2-fold cross validation while the leftmost points result from using 11-folds. Due to the small training sets, interpreting the figures is difficult. The unsupervised baseline outperforms the learning methods although there are indications that more training data could lead to better performances of perceptrons and SVMs. The first feature representation shows better performances than the second. However, these conjectures need to be verified by an experiment on a larger scale.

Finally, Table 1 shows average accuracies and average ranks for a leave-one-out setup using the third feature representation. The results are promising and not too far from the baseline, however, as before, the evaluation needs to be based on larger sample sizes to allow for interpretations.

5 Related work

Barzilay et al. [9] study sentence compression using dependency trees. Aligned trees are represented by a lattice from which a compression sentence is extracted by an entropy-based criterion over all possible traversals of the lattice. Wan et al. at [11] use a language models in combination with maximum spanning trees to rank candidate aggregations that satisfy grammatical constrains.

While the previous approaches to multi-sentence compression are based on syntactic parsing of the sentences, word graph approaches have been proposed, that do not make use of dependency trees or other linguistic concepts. Filippova [4] casts the problem as finding the shortest path in directed word graphs, where each node is a unique word and directed edges represent the word ordering in the original sentences. The costs of these edges are a heuristic function that is based on word frequencies. Recently, Boudin and Morin [10] propose a re-ranking scheme to identify summarising sentences that contain many keyphrases. The underlying idea is that representative key phrases for a given topic give rise to more informative aggregations.

In contrast to the cited related work, we cast the problem of sentence compression as a supervised structured prediction problem and aim at learning the edge costs from a possibly rich set of features describing adjacent nodes.

6 Conclusion

In this paper, we proposed to learn shortest paths in compression graphs for summarising related sentences. We addressed the previously unsupervised problem in a supervised context and devised structured perceptrons and support vector machines that effectively learn the edge weights of compression graphs, so that a shortest path algorithm decodes the best possible summarisation. We showed that the most strongly violated constrains can be computed directly by loss-augmented inference and rendered the use of expensive two-best algorithms unnecessary. Empirically, we presented preliminary results on artificial and real world data sets. Due to small sample sizes, conclusions cannot be confidently drawn yet, although the results seemingly indicate that learning shortest paths could be an alternative to heuristic and unsupervised approaches. Future work will address this question in greater detail.

References

1. U. Brefeld. Cost-based Ranking in Input Output Spaces. Proceedings of the Workshop on Learning from Non-vectorial Data, 2007.
2. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research, 6 (Sep):1453-1484, 2005
3. R. Barzilay, L. Lee, Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment, in Proc. of NAACL-HLT, 2003.
4. K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs, COLING, 2010
5. J. Y. Yen, Finding the k Shortest Loopless Paths in a Network. Management Science 17 (11): 712716, 1971
6. R. Bellman (1958). "On a routing problem". Quarterly of Applied Mathematics 16: 8790. MR 0102435.
7. J. Ford, R. Lester (August 14, 1956). Network Flow Theory. Paper P-923. Santa Monica, California: RAND Corporation.

8. E. W. Dijkstra (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269271. doi:10.1007/BF01386390.
9. R. Barzilay and K. McKeown. Sentence Fusion for Multidocument News Summarization, Computational Linguistics, 2005.
10. F. Boudin and E. Morin. Keyphrase Extraction for N-best Reranking in Multi-Sentence Compression, Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013.
11. S. Wan, R. Dale, M. Dras, C. Paris. Global revision in summarisation : generating novel sentences with Prim's algorithm, Conference of the Pacific Association for Computational Linguistics, 2007.
12. B. Taskar and D. Klein and M. Collins and D. Koller and C. Manning. Max-margin parsing, Proc. EMNLP, 2004.
13. M. Collins and N. Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron, ACM, 2002
14. Y. Altun, M. Johnson, T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences, Proc. EMNLP, 2003.
15. Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu
16. Leipzig Corpora Collection (LCC). Universität Leipzig, Institut für Informatik, Abteilung Sprachverarbeitung. http://corpora.uni-leipzig.de/