Learning to Summarise Related Sentences

Emmanouil Tzouridis[†]
[†] Dep. of Computer Science
TU Darmstadt
Germany
{tzouridis,brefeld}@kma.informatik.tu-darmstadt.de

Jamal Abdul Nasir
Dep. of Computer Science
LUMS Lahore
Pakistan
jamaln@lums.edu.pk

Ulf Brefeld[†‡]
[‡] Inform. Center f. Education
DIPF Frankfurt/Main
Germany
brefeld@dipf.de

Abstract

We cast multi-sentence compression as a structured prediction problem. Related sentences are represented by a word graph so that summaries constitute paths in the graph (Filippova, 2010). We devise a parameterised shortest path algorithm that can be written as a generalised linear model in a joint space of word graphs and compressions. We use a large-margin approach to adapt parameterised edge weights to the data such that the shortest path is identical to the desired summary. Decoding during training is performed in polynomial time using loss augmented inference. Empirically, we compare our approach to the state-of-the-art in graph-based multi-sentence compression and observe significant improvements of about 7% in ROUGE F-measure and 8% in BLEU score, respectively.

1 Introduction

Automatic text summarisation is one of oldest forms of natural language processing (Luhn, 1958; Baxendale, 1958). The goal is to extract the most important part of the content from either a single document or a collection of documents (Mani, 2001; Roussinov and Chen, 2001; McKeown *et al.*, 2005).

Frequently, the information of interest is contained in only a part of a sentence or may be distributed across parts of several sentences. Identifying the content carrying part(s) constitutes an essential technique not only for single- and multi-document extractive summarisation but also text simplification in general. Generating a simplified version of a text traditionally has many applications in question answering (Hermjakob *et al.*, 2002) and speech synthesis (Kaji *et al.*, 2004). Due to limited display sizes of mobile devices, recent applications also deal with summarising/simplifying news articles, social media, emails, or websites (Corston-Oliver, 2001).

Multi-sentence compression (MSC) unifies many of the mentioned characteristics and challenges and can be seen as a key to text summarisation and simplification (Jing and McKeown, 2000). The task in multi-sentence compression is to map a collection of related sentences to a grammatical short sentence that preserves the most important part of the content. Sentence compression methods have been devised using manually crafted rules (Dorr *et al.*, 2003), language models (Hori *et al.*, 2003; Clarke and Lapata, 2008), or syntactical representations (Barzilay and Lee, 2003; Galley and McKeown, 2007; Filippova and Strube, 2008). Filippova (2010) introduces an elegant graph-based approach to multi-sentence compression that simply relies on the words of the sentences and efficient dynamic programming. Her approach implements the observation that the frequency of words influences their appearance in human summaries (Nenkova *et al.*, 2006). Although being an intuitive rule that does work well in practice, frequency-based strategies often remain heuristic.

In this paper we propose a structured learning-based approach to multi-sentence compression. In analogy to Filippova (2010), related sentences are represented by a word graph (the *input*). Words are identified with vertices and directed edges connect adjacent words in at least one sentence, so that the summarising sentence (the *output*) is contained as a path in the graph. Generally, learning mappings between complex structured and interdependent inputs and outputs challenges the standard model of

learning a mapping from independently drawn instances to a small set of labels. To capture the involved dependencies we represent input graphs $\mathcal{G}$ and output paths p jointly by a (possibly rich) feature representation $\Phi(\mathcal{G}, p)$. The goal is to find a linear function $f(\mathcal{G}, p) = \lambda^\top \Phi(\mathcal{G}, p)$ in joint space such that

$$p = \operatorname*{argmin}_{\tilde{p}} f(\mathcal{G}, \tilde{p}) \tag{1}$$

is the desired summary for the collection $\mathcal{G}$. Our approach can therefore be seen as translating the work by Filippova (2010) into the structured prediction framework (Tsochantaridis *et al.*, 2005; Taskar *et al.*, 2004). Instead of applying heuristics, we adapt the decoding machinery to the data by parameterising a shortest path algorithm. The latter admits a representation as a generalised linear model in joint input output space. We devise a structural support vector machine (SVM) (Tsochantaridis *et al.*, 2005) to learn the shortest path in possibly high dimensional joint feature spaces and propose a generalised, loss-augmented decoding algorithm that is solved exactly by an integer linear program in polynomial time. Empirically, we evaluate the structural support vector machine on a real world news headline summarisation task. Our experiments show that a very rudimentary set of five features already suffices to significantly improve the state-of-the-art in graph-based multi-sentence compression. We observe an increase of 7% in in ROUGE F-measure and 8% in BLEU score, respectively.

The remainder of the paper is organised as follows. Section 2 reviews related work and Section 3 introduces word graphs and shortest paths. Our technical contribution is presented in Section 4. We report on empirical results in Section 5 and Section 6 concludes.

2 Related Work

The goal of automatic text summarisation is to produce a summary of a given text (or text collection) that preserves the most important information (Luhn, 1958; Edmundson, 1969). Summarisation systems usually rely on clues or features that help to identify key elements such as the main topic of a document (Salton *et al.*, 1994). Such features may be extracted from sentences (e.g., the length of a sentence, its position in the text), words (e.g., frequency of a word, relative position in sentence) as well as from style and structure elements (Kupiec *et al.*, 1995; Teufel and Moens, 1997; Marcu, 1997).

A special case of text summarisation is sentence compression; given a sentence, the task is to produce a summary of the input that preserves the most important information and is grammatically correct (Jing, 2000). Sentence compression is thus relevant for many NLP tasks including question answering, machine translation, text simplification, speech synthesis applications and multi-sentence compression (e.g., Lin (2003)).

Multi-sentence compression extends sentence compression to collections of related sentences that are to be summarised in a single output sentence. Traditionally, contributions to multi-sentence compression exploit linguistic properties based on lexical information and syntactic dependencies. Dorr *et al.* (2003) for instance propose a headline generation system based on linguistically-motivated, hand-crafted heuristics. Barzilay and Lee (2003) study sentence compression with dependency trees. The aligned trees are represented by a lattice from which a summary is extracted by an entropy-based criterion over all possible traversals of the lattice. Similarly, Barzilay and McKeown (2005) combine syntactic trees of similar sentences by a multi-sequence alignment candidate selection and summary generation. Wan (2007) deploys a language model in combination with maximum spanning trees to rank candidate aggregations satisfying grammatical constraints. Hori *et al.* (2003) propose a statistical model for automatic speech summarisation without using parallel data or syntactic information. Instead they focus on language models to provide a scoring function and use dynamic programming for searching the compression with the highest score. Clarke and Lapata (2008) cast sentence compression as an optimisation problem. They use linguistically motivated constraints and integer linear programming to infer globally optimal compressions.

Recently, graph-based approaches to multi-sentence compression have been proposed. The underlying idea is that syntax may help to find important content. Thus, instead of using hand-crafted rules, parsers, or language models, a simple and robust graph-based method can be used to generate reasonable summaries. Graph-based multi-sentence compression approaches identify the summary with the

shortest path in word graphs (Filippova, 2010). Shortest paths of unweighted word graphs however do not necessarily lead to satisfying summaries. As a remedy, Filippova (2010) introduces heuristic edge weights based on normalised frequencies of the connected words. Boudin and Morin (2013) propose an additional re-ranking scheme to identify summarisations that contain key phrases. The underlying idea is that particular key phrases give rise to certain topics and thus lead to more informative aggregations.

In this paper, we parameterise the graph-based framework by Filippova (2010) such that the shortest path algorithm is adapted to labeled data at hand. Adapting the dynamic programming to the data renders the use of heuristics unnecessary. Instead, word graphs and compressions are embedded in a (possibly high-dimensional) joint feature space where a generalised linear scoring function learns to separate between compressions of different quality. We develop a generalised, loss-augmented shortest path algorithm that is solved exactly by a (relaxed) integer linear program in polynomial time.

3 Preliminaries

3.1 Word Graphs

In a nutshell, word graphs represent collections of sentences efficiently in a graph by mapping identical words to a single vertex while the graph structure preserves the local neighbourhood of words.

From a collection of related sentences a word graph is constructed as follows: Initially, every sentence is augmented by a preceding start token $\langle S \rangle$ and a terminal end symbol $\langle E \rangle$ so that beginning and end of the sentences are preserved in the final graph. Starting with the empty graph, sentences are added one after another. The first word of the first sentence is the auxiliary $\langle S \rangle$ that is converted into the first vertex $v_{\langle S \rangle}$. The second word of the first sentence also becomes a vertex v and the two vertices are connected with a directed edge $v_{\langle S \rangle} \to v$. The procedure continues with the third word and so on until the end symbol $\langle E \rangle$ is reached. The other sentences are incorporated analogously. A special case arises if the graph already contains a vertex v that is identical to the word that is just to be added. Instead of adding a redundant vertex, the already existing vertex v is used and, if $v \neq v_{\langle S \rangle}$, connected to the respective predecessor as before. In that case, the vertex v has an in-degree of (at least) two and is used as the predecessor for the next word to be added. The procedure continues until all n sentences are incorporated in the graph.

Note that merging nodes to the same vertex requires an appropriate preprocessing of the sentences. Simple lower- or upper-case representations of words often suffice but more complex preprocessing schemas are also possible such as merging vertices carrying synonyms or words possessing small WordNet distances (Miller, 1995; Fellbaum, 1998). As word graphs are a condensed representation of the input sentences, word graphs are also known as compression graphs. The described construction gives us a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of unique words in the sentences and $\mathcal{E}$ the set of neighbouring words. An exemplary word graph is shown in Figure 1.

3.2 Shortest Path Algorithms

Given a directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, cost)$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges. The function $cost : (v, v') \mapsto \Re^+$ assigns positive weights to every edge $(v, v') \in \mathcal{E}$. A path p from a vertex $v_s \in \mathcal{V}$ to a vertex $v_e \in \mathcal{V}$ is a sequence of edges connecting vertices of $\mathcal{G}$. We write $\mathcal{P}(v_s, v_e)$ to denote the set of all possible paths starting in v_s and terminating in v_e. The cost of a path is given by the sum of the weights of the edges on the path.

The *shortest path* from a start vertex $v_s \in \mathcal{V}$ to an end vertex $v_e \in \mathcal{V}$ is defined as the path in G from v_s to v_e with the lowest costs. Introducing auxiliary binary variables $p_{(v,v')}$ indicating whether an edge $(v, v') \in \mathcal{E}$ lies on the path ($p_{v,v'} = 1$) or not ($p_{v,v'} = 0$) the shortest path can be computed by the following optimisation problem

$$p^* = \operatorname*{argmin}_p \sum_{(v,v') \in \mathcal{E}} p_{v,v'} \, cost(v, v') \quad \text{s.t.} \quad p \in \mathcal{P}(v_s, v_e). \tag{2}$$

There exist many algorithms for computing shortest paths efficiently (Bellman, 1958; Ford, 1956; Dijkstra, 1959). Usually, these methods are based on dynamic programming or (relaxed) integer program-

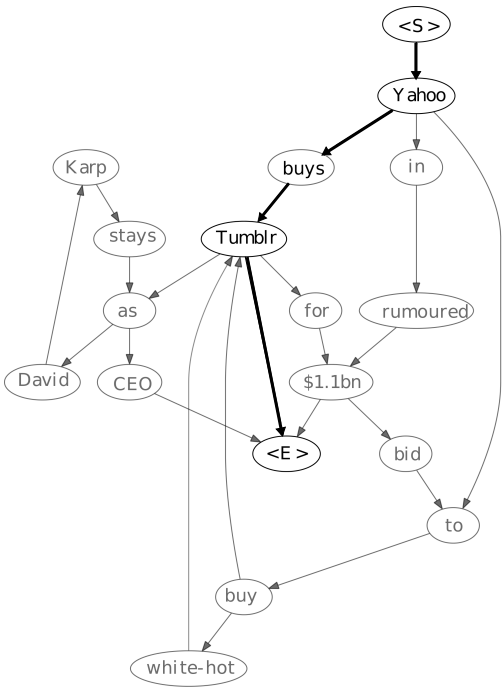

Figure 1: The word graph constructed from the sentences: "Yahoo in rumoured $1.1bn bid to buy white-hot Tumblr", "Yahoo buys Tumblr as David Karp stays as CEO", "Yahoo to buy Tumblr for $1.1bn". The shortest path is highlighted.

ming, where an approximation of the exact quantity is iteratively updated until it converges to the correct solution, which is achieved in polynomial time. A prominent algorithm for computing the k-shortest paths is Yen's algorithm (Yen, 1971). 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.

4 Learning to Summarise Related Sentences

4.1 Problem Setting

Given a word graph $\mathcal{G}$, we aim to find a ranking function $f(\mathcal{G}, p)$ that assigns the lowest score to the best summary p^*, that is, $p^* \stackrel{!}{=} \operatorname{argmin}_p f(\mathcal{G}, p)$. Note that f is defined jointly on $\mathcal{G}$ and p to allow for exploiting dependencies between word graph and summary. Our approach can thus be seen as an instance of structured prediction models. The quality of f is measured by the Hamming loss Δ, $\Delta(p^*, \hat{p}) = \frac{1}{2}\sum_{(v_i,v_j)\in\mathcal{V}}[[p^*_{ij} \neq \hat{p}_{ij}]]$, that details differences between the best summary p^* and the prediction $\hat{p}$, where $[[z]]$ is the indicator function returning one if z is true and zero otherwise. The generalisation error is given by

$$R[f] = \int \Delta\left(p, \operatorname*{argmin}_{\tilde{p}} f(\mathcal{G}, \tilde{p})\right) dP(\mathcal{G}, p)$$

and approximated by its empirical counterpart

$$\hat{R}[f] = \sum_{i=1}^{m} \Delta\left(p_i, \operatorname*{argmin}_{\tilde{p}} f(\mathcal{G}_i, \tilde{p})\right) \tag{3}$$

on a finite m-sample of pairs $\{(\mathcal{G}_i, p_i)\}_{i=1}^m$ where $\mathcal{G}_i$ is a word graph and p_i the best summarising sentence. However, minimising the empirical risk directly leads to an ill-posed optimisation problem as

there generally exist many indistinguishable but equally well solutions realising an empirical loss of zero. We thus focus on the minimisation of the regularised empirical risk

$$\hat{R}_{reg}[f] = \Omega(f) + \sum_{i=1}^{m} \Delta \left(p_i, \operatorname*{argmin}_{\tilde{p}} f(\mathcal{G}_i, \tilde{p}) \right).$$

The additive regularisation $\Omega(f)$ acts like a prior on f, e.g. to enforce smooth solutions. In the remainder we use $\Omega(f) = \|f\|^2$.

4.2 Representation

The idea of our approach is as follows: We adapt the cost function of the graph to the training sample such that the shortest path of the compression graph is identical to the desired summary. Recall the general form of the cost function of Section 3.2. Instead of a constant or hand-crafted function (Filippova, 2010), we deploy a linear mixture of features ϕ_i, parameterised by λ,

$$cost(v, v') = \sum\nolimits_i \lambda_i \phi_i(v, v') = \lambda^\top \phi(v, v').$$

Features $\phi_i(v, v')$ are drawn from adjacent vertices v, v' in the word graph to capture local dependencies of the connecting edge. Examples for feature functions are frequency-based counts or indicators such as POS-transitions of the form $\phi_{234}(v, v') = [[v \text{ is a noun} \wedge v' \text{ is a verb}]]$. Note that complex features using the context of the edge are straight forward by extending the feature representation to the input graph $\phi(v, v', \mathcal{G})$. The final feature vector is obtained by stacking-up all feature functions, that is, $\phi(v, v') = (\ldots, \phi_i(v, v'), \ldots)^\top$.

Using the parameterised costs in the computation of the shortest path in Equation (2) yields the following objective function (ignoring the constraints for a moment) that can be rewritten as a generalised linear model in joint input output space

$$\sum_{(v_i,v_j)\in\mathcal{V}} p_{ij}\, \lambda^\top \phi(v_i, v_j) = \lambda^\top \underbrace{\left(\sum_{(v_i,v_j)\in\mathcal{V}} p_{ij} \phi(v_i, v_j) \right)}_{\equiv \Phi(\mathcal{G}, p)} = \lambda^\top \Phi(\mathcal{G}, p) = f(\mathcal{G}, p)$$

where the joint feature representation is given by

$$\Phi(\mathcal{G}, p) \equiv \left(\sum_{(v_i,v_j)\in\mathcal{V}} p_{ij} \phi(v_i, v_j) \right).$$

Decoding the shortest path $\hat{p}$ for a fixed parameter vector λ can now be computed by

$$\hat{p} = \operatorname*{argmin}_{p} f(\mathcal{G}, p) \quad s.t.\ \ p \in \mathcal{P}(\langle S \rangle, \langle E \rangle)$$

using standard shortest path algorithms (Yen, 1971). In addition, reformulating the objective as a generalised linear model allows to adapt the parameters λ to the data to identify shortest paths with summaries.

4.3 Optimisation

Recall that the goal of the optimisation is to find the ranking function $f(\mathcal{G}, p)$ that takes the smallest value for the best summary. That is, for the i-th training instance $(\mathcal{G}_i, p_i)$, we aim at fulfilling the constraints

$$\lambda^\top \Phi(\mathcal{G}_i, p) - \lambda^\top \Phi(\mathcal{G}_i, p_i) > 0 \tag{4}$$

for all $p \in \mathcal{P}(\langle S \rangle, \langle E \rangle) \backslash p_i$. We extend the constraints in Equation (4) by a term that induces a margin between the best path p_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 $\tilde{p}$ instead of p_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 constraints

$$\lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) - \lambda^\top \Phi(\mathcal{G}_i, p_i) > \Delta(p_i, \tilde{p}) - \xi_i$$

for all $p \in \mathcal{P}(\langle S \rangle, \langle E \rangle) \backslash p_i$. The non-negative $\xi_i \geq 0$ is a slack-variable that allows point-wise 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,

$$\ell_\Delta(\mathcal{G}_i, p_i, f) = \max \left[\min_{\tilde{p}} \left[\Delta(p_i, \tilde{p}) - f(\mathcal{G}_i, \tilde{p}) + f(\mathcal{G}_i, p_i) \right] \right].$$

The effective hinge loss upper bounds the structural loss Δ for every pair $(\mathcal{G}_i, p_i)$ and trivially also

$$\sum_{i=1}^{m} \ell_\Delta(\mathcal{G}_i, p_i, f) \geq \sum_{i=1}^{m} \Delta(p_i, \operatorname*{argmin}_{\tilde{p}} f(\mathcal{G}_i, \tilde{p}))$$

holds. A max-margin approach to learning shortest paths therefore leads to the following optimisation problem that is also known as structural support vector machine (Tsochantaridis *et al.*, 2005)

$$\min_{\lambda, \xi \geq 0} \ \|\lambda\|^2 + C \sum_{i=1}^{m} \xi_i \quad s.t. \quad \forall i \, \forall \tilde{p} \in P \backslash p_i : \ f(\tilde{p}) - f(p_i) > \Delta(p_i, \tilde{p}) - \xi_i. \tag{5}$$

The parameter C trades-off margin maximisation and error minimisation and needs to be adjusted by the user. The above optimisation problem can be solved efficiently by cutting plane methods. The idea behind cutting planes is to instantiate only a minimal subset of the exponentially many constraints. That is, for the i-th training example, we decode the shortest path $\hat{p}$ given our current model and consider two cases: (i) For $\hat{p} \neq p_i$ the prediction is erroneous and $\hat{p}$ is called the most strongly violated constraint as it realises the smallest function value and $f(\mathcal{G}_i, \hat{p}) < f(\mathcal{G}_i, p)$ holds for all $p \neq \hat{p}$. 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 $\hat{p} = p_i$, we need to verify that the runner-up $\hat{p}^{(2)}$ 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

$$Q(\tilde{p}) = \Delta(p_i, \tilde{p}) - \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) + \lambda^\top \Phi(\mathcal{G}_i, p_i) \tag{6}$$

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

Proposition 1. *The argmax $\tilde{p}$ of Q in Equation* (6) *can be computed by minimising a shortest path problem with cost function $cost(v_i, v_j) = p_{ij} + \lambda^\top \phi(v_i, v_j)$.*

Proof. We treat the ground truth paths p as graphs and write $\mathcal{V}(p)$ for the set of nodes on the path and $\mathcal{E}(p)$ to denote the set of edges that lie on the path. If, for instance, an element of the binary adjacency matrix representing path p equals one, e.g., $p_{ij} = 1$, we write $p_i, p_j \in \mathcal{V}(p)$ and $(p_i, p_j) \in \mathcal{E}(p)$. First, note that the Hamming loss can be rewritten as

$$\Delta(p, \tilde{p}) = \sum_{(p_i, p_j) \in \mathcal{E}(p)} (1 - p_{ij} \tilde{p}_{ij}) \, .$$

We have

$$\begin{aligned}
\hat{p} &= \operatorname*{argmax}_{\tilde{p}} \Delta(p,\tilde{p}) + \lambda^\top \Phi(\mathcal{G}_i, p) - \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) \\
&= \operatorname*{argmax}_{\tilde{p}} \Delta(p,\tilde{p}) - \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) \\
&= \operatorname*{argmax}_{\tilde{p}} \sum_{(p_i,p_j)\in\mathcal{E}(p)} (1 - p_{ij}\tilde{p}_{ij}) - \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) \\
&= \operatorname*{argmax}_{\tilde{p}} -\sum_{(p_i,p_j)\in\mathcal{E}(p)} p_{ij}\tilde{p}_{ij} - \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) \\
&= \operatorname*{argmin}_{\tilde{p}} \sum_{(p_i,p_j)\in\mathcal{E}(p)} p_{ij}\tilde{p}_{ij} + \lambda^\top \Phi(\mathcal{G}_i, \tilde{p}) \\
&= \operatorname*{argmin}_{\tilde{p}} \sum_{(p_i,p_j)\in\mathcal{E}(p)} p_{ij}\tilde{p}_{ij} + \lambda^\top \Big[\sum_{(x_i,x_j)\in E(\mathcal{G})} \tilde{p}_{ij}\phi(v_i,v_j)\Big] \\
&= \operatorname*{argmin}_{\tilde{p}} \sum_{(v_i,v_j)\in\mathcal{E}(\mathcal{G})} p_{ij}\tilde{p}_{ij} + \lambda^\top \Big[\sum_{(x_i,x_j)\in E(\mathcal{G})} \tilde{p}_{ij}\phi(v_i,v_j)\Big] \\
&= \operatorname*{argmin}_{\tilde{p}} \sum_{(v_i,v_j)\in\mathcal{E}(\mathcal{G})} \Big[p_{ij} + \lambda^\top \phi(v_i,v_j)\Big]\tilde{p}_{ij}
\end{aligned}$$

The output $\hat{p}$ is the shortest path with costs given by $p_{ij} + \lambda^\top \phi(v_i, v_j)$. □

Using this result, the following lemma shows that we can compute the most strongly violated constraint directly by a linear program.

Lemma 1. *The maximizer $\tilde{p}$ of function Q in Equation* (6) *and thus the shortest path of Proposition 1 can be computed in polynomial time by the following linear program*

$$\min_{\tilde{p}} \sum_{ij} \Big(p_{ij} + \lambda^\top \phi(v_i,v_j)\Big)\,\tilde{p}_{ij}$$

subject to the constraints

$$\begin{aligned}
\forall k \in \mathcal{V}(\mathcal{G})\backslash\{\langle S\rangle, \langle E\rangle\} :\ \sum_j \tilde{p}_{kj} - \sum_i \tilde{p}_{ik} \le 0 \ \wedge\ -\sum_j \tilde{p}_{kj} + \sum_i \tilde{p}_{ik} \le 0 \\
\sum_j \tilde{p}_{\langle S\rangle,j} - \sum_i \tilde{p}_{i,\langle S\rangle} \le 1 \ \wedge\ -\sum_j \tilde{p}_{\langle S\rangle,j} + \sum_i \tilde{p}_{i,\langle S\rangle} \le -1 \\
\sum_i \tilde{p}_{i,\langle E\rangle} - \sum_j \tilde{p}_{\langle E\rangle,j} \le 1 \ \wedge\ -\sum_i \tilde{p}_{i,\langle E\rangle} + \sum_j \tilde{p}_{\langle E\rangle,j} \le -1 \\
\forall (i,j) :\ \tilde{p}_{ij} \le \mathcal{G}_{(i,j)} \ \wedge\ \forall (i,j) :\ \tilde{p}_{ij} \in \{0,1\}.
\end{aligned}$$

Proof. For lack of space, we only motivate the constraints. The first line of constraints guarantees that every inner node of the path has exactly as many incoming as outgoing edges, the second line forces the path to start in $v_{\langle S\rangle}$ and, analogously, the third line ensures that it terminates in $v_{\langle E\rangle}$. The last line of constraints requires the edges of the path $\tilde{p}$ to adhere to existing paths of $\mathcal{G}$. □

4.4 Parallelisation

Using the result by Zinkevich *et al.* (2011) the proposed approach can easily be distributed on several machines. Note that cutting planes treat one input $(\mathcal{G}, p)$ at a time. Thus, several models can be trained independently in parallel on disjoint subsets of the data. A subsequent merging process aggregates the models where each models impact is proportional to the amount of data it has been trained on. Note that the described parallelisation can easily be realised by the MapReduce/Hadoop framework. Processing training instances and updating local models is performed by (one or more) mappers while the merge operation is carried out by a reduce task.

Table 1: Left: Collection of related sentences. Right: Candidate compressions and number of votes.

related sentences
White House: Hong Kong had 'plenty of time' to stop Snowden live coverage
Edward Snowden leaves reporters chasing shadows around an airport
US warns Moscow not to let Edward Snowden escape Russia
WikiLeaks forced to defend Ecuador as Edward Snowden seeks asylum
Snowden is 'not on plane' to Cuba

summary	#
snowden seeks asylum	5
snowden live coverage	5
snowden escape russia	1
edward snowden seeks asylum	3
wikileaks forced to cuba	1

5 Empirical Results

5.1 Data Preparation

We crawl RSS feeds of 6 major news sites and harvest news headlines of a predefined set of categories including sports, technology, and business. The headlines are processed automatically by a spectral clustering. The data is thus transformed into a fully connected graph where vertices correspond to headlines and edges are weighted by the number of shared non-stopwords. The clustering is performed for each category on a daily basis. Resulting clusters are headlines that belong (with high probability) to the same event and form our related input sentences. Groups with less than five sentences are discarded.

To identify the best summaries, we conduct a crowd sourcing experiment on Crowdflower[1]. Every annotator is given a group of related sentences together with 10 possible summaries generated by a 10-best Yen's algorithm (Yen, 1971). The task of the annotator is to pick the best summary or mark the collection as inappropriate. Each collection is labeled by at least 10 annotators. The group is discarded if the majority of the annotators mark the group as inappropriate. Otherwise, the three most frequent summaries are extracted, ties are broken by the authors. The most frequent summary is used as the ground-truth annotation in the learning phase, the other two are used additionally in the evaluation. The described process leaves us with 1024 sentences that are divided into 164 annotated groups of related sentences. Table 1 shows an exemplary collection of related sentences (left) and a selection of summaries together with the number of votes from the annotators. The overall distribution of votes is displayed in Figure 2. The figure shows the mean value per rank of all 164 normalised and sorted histograms. The best summary receives on average 8% more votes than the runner-up (not shown).

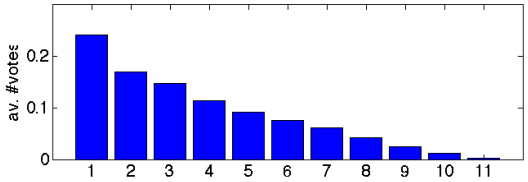

Figure 2: Distribution of annotations.

5.2 Baselines and Features

We compare our learning approach to graph-based sentence compression techniques proposed by Filippova (2010), Boudin and Morin (2013). The two baselines construct word graphs as presented in Section 3.1 and output the weighted shortest path. Filippova (2010) uses a frequency-based heuristic for edges weights and Boudin and Morin incorporate a keyphrase detection framework to re-rank summaries according to the number and importance of keyphrases found. In addition, we also include an unweighted shortest path strategy which is a straight forward application of Yen's algorithm (Yen, 1971) and trivially returns the shortest path in terms of the number of edges. Additional straw men are a random (Random) input sentence and the shortest input sentence (Shortest).

In our learning-based approach, every edge between vertices v and v' is associated with a feature vector. Let $w = \#(v)$ the frequency of word v, w' the analogue for v', $e = \#(v, v')$ the frequency of the edge, and $n = |\mathcal{V}|$ the number of vertices in the graph. The feature vector $\phi(v, v')$ of

[1] http://crowdflower.com

Table 2: ROUGE F-measure scores

		training set size						
		22	35	48	61	74	87	100
R1	Random	46.72	46.82	46.41	46.20	46.39	46.53	46.88
	Shortest	45.93	45.77	46.39	46.56	47.01	47.59	48.04
	Yen	45.14	44.47	45.12	45.13	45.63	46.14	46.39
	Filippova	52.70	52.94	52.16	52.02	52.22	52.45	51.81
	Boudin	52.72	53.12	53.43	53.52	53.10	52.81	52.35
	SVM	48.39	50.30	**55.09**	**54.59**	**57.39**	**54.89**	**57.66**
R2	Random	30.43	30.63	30.56	30.31	30.38	30.64	31.09
	Shortest	27.65	27.43	27.90	27.93	28.64	29.47	30.10
	Yen	31.38	30.82	31.16	31.40	31.90	32.30	32.56
	Filippova	36.12	36.52	35.56	35.49	35.75	35.98	35.64
	Boudin	36.71	37.01	37.79	37.65	36.97	36.75	36.31
	SVM	33.64	35.40	**40.46**	**40.68**	**43.44**	**40.45**	**43.58**
RW1.2	Random	35.91	35.97	35.74	35.58	35.80	35.93	36.07
	Shortest	34.47	34.29	34.77	34.85	35.32	35.9	36.16
	Yen	34.85	34.26	34.74	34.83	35.22	35.62	35.77
	Filippova	40.30	40.53	39.88	39.70	39.94	40.12	39.56
	Boudin	40.79	40.99	41.37	41.31	40.92	40.83	40.36
	SVM	37.94	39.06	**42.61**	**42.33**	**44.63**	**42.90**	**45.00**

the edge $v \rightarrow v'$ consists of the normalised joint frequency $\phi_1(w, w') = \frac{e}{n}$, the maximal word frequency $\phi_2(w, w') = \max\left\{\frac{w}{n}, \frac{w'}{n}\right\}$, the lexical relevance $\phi_3(w, w') = \frac{2}{n}\frac{w \cdot w'}{w+w'}$, the normalised PMI $\phi_4(w, w') = (\log \frac{e}{w \cdot w'})/ - \log \frac{e}{n}$ (Bouma, 2009), and ϕ_5 captures the average location of the phrase in the input sentences (Turney, 2000),

$$\phi_5(w, \tilde{w}) = \begin{cases} 1.0 & : & [0-10]\% \\ 0.4 & : & [10-30]\% \\ 0.8 & : & [30-60]\% \\ 0.6 & : & [60-80]\% \\ 1.0 & : & [80-100]\%. \end{cases}$$

Note that $\phi_i \in [0, 1]$ holds for $1 \leq i \leq 5$. Also note that ϕ denotes a rudimentary set of features. Elaborate representations could for instance also contain POS-tags or named entities.

5.3 Experimental Setup and Results

For the news headline experiment, we draw $m \in \{22, 35, 48, 61, 74, 87, 100\}$ training instances without replacement at random from the collected data. The remaining instances are split randomly into equally sized holdout and test sets. We perform model selection for adjusting the trade-off parameter of the support vector machine on the interval $C \in [2^{-10}, 2^{12}]$. We report average ROUGE F-measures (Lin, 2004) and BLEU scores (Papineni *et al.*, 2012) over 10 repetitions with distinct training, holdout, and test sets. In each repetition, all algorithms are trained and/or evaluated on identical data splits.

ROUGE measures the concordance of system and human generated summaries by determining n-gram, word sequence, and word pair matches. We use unigrams (R1), bigrams (R2), and the weighted longest common subsequence (RW1.2) to evaluate compressions. Note that R1 has been found to correlate well with human evaluations based on various statistical metrics (Lin and Hovy, 2003). Moreover, R1 and R2 emulate human pyramid and responsiveness scores (Owczarzak *et al.*, 2012).

Table 2 shows the resulting ROUGE scores for the news headline experiment. Significant results are marked in bold face according to a paired t-test using a significance level of 5%. For small training sets, the structural support vector machine performs only slightly better than the unweighted application of Yen's algorithm and is clearly outperformed by the unsupervised baselines. However, the SVM improves

Table 3: BLEU scores

		training set size						
		22	35	48	61	74	87	100
B1	Random	38.56	38.40	36.49	36.77	36.00	36.87	37.35
	Shortest	37.45	38.37	38.46	37.25	37.28	37.17	36.64
	Yen	29.46	28.3	29.39	29.98	29.99	31.20	30.64
	Filippova	44.26	43.29	44.66	44.57	45.21	43.10	43.52
	Boudin	44.00	42.54	44.75	44.39	44.80	43.22	43.96
	SVM	39.60	41.96	**48.44**	**47.10**	**50.20**	**46.90**	**50.39**
B2	Random	34.85	34.80	33.12	33.48	32.65	33.77	34.12
	Shortest	33.34	34.27	34.54	33.39	33.39	33.43	33.45
	Yen	28.51	27.27	28.34	28.74	29.06	30.05	29.73
	Filippova	39.92	39.36	40.05	40.27	41.14	39.26	39.60
	Boudin	39.43	38.45	39.99	40.02	40.52	39.20	39.84
	SVM	36.37	38.63	**45.31**	**44.15**	**47.40**	**43.75**	**47.44**
B3	Random	35.91	35.97	35.74	35.58	35.80	35.93	36.07
	Shortest	34.47	34.29	34.77	34.85	35.32	35.90	36.16
	Yen	27.85	26.64	27.61	27.84	28.38	29.34	28.93
	Filippova	36.07	35.88	35.86	36.42	37.37	35.55	35.97
	Boudin	35.05	34.39	35.40	35.76	36.17	34.93	35.85
	SVM	33.26	35.39	**42.31**	**41.05**	**44.54**	**40.52**	**44.51**

continuously with increasing training set sizes and outperforms the baselines significantly for more than 50 training examples. The unsupervised baselines cannot utilise the valuable annotations of the data and remain constant. For 100 training instances, we observe performance improvements of about 5-7% for all three ROUGE F-measures.

The BLEU metric computes scores for individual segments, then averages these scores over the whole corpus for a final score. For our experiments we use BLEU-1, BLEU-2 and BLEU-3 to evaluate compressions. Table 3 shows the corresponding results, significant results are again marked in bold face according to a paired t-test with a significance level of 5%. The table draws a similar picture than the previous one. The SVM continuously improves the performance with increasing training set sizes and beats the baselines again at about 50 training examples significantly. For 100 training instances, all three BLEU scores are improved by about 7-8%, respectively.

5.4 Analysis

The Pearson correlation between BLEU scores per instance and the number of vertices is -0.1886. The negative correlation implies that summarising larger word-graphs is more challenging. A negative correlation of -0.1267 is also observed for the lexical diversity of the collection; diverse groups of sentences are thus more difficult to summarise. A possible remedy could be features that are not frequency-based, such as POS-transitions. By contrast, the density of the graph given by $|\mathcal{E}|/|\mathcal{V}|(|\mathcal{V}|-1)$ shows a positive correlation of 0.1851. The more dense a graph, the more edges interconnect vertices and there exist more paths. These paths however frequently pass through the same vertices and as a consequence the lexical diversity is low. A positive correlation of graph density is therefore closely connected to a negative correlation of lexical diversity.

6 Conclusion

We proposed to learn shortest paths in word graphs for multi-sentence compression. A shortest path algorithm is parameterised and adapted to labeled data at hand using the structured prediction framework. Word graphs and summaries are embedded in a joint feature space where a generalised linear scoring function learns to separate between compressions of different quality. Decoding is performed

by a generalised, loss-augmented shortest path algorithm that can be solved by an integer linear program in polynomial time. Empirically, we observe that a very rudimentary set of five features suffices to significantly improve the state-of-the-art in graph-based multi-sentence compression.

Acknowledgments

Jamal Abdul Nasir is supported by a grant from the Higher Education Commission, H-9 Islamabad, Pakistan.

References

R. Barzilay and L. Lee. 2003. *Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment*, Proceedings of NAACL-HLT.

R. Barzilay and K. R. McKeown. 2005. *Sentence Fusion for Multidocument News Summarization*, Comput. Linguist. 31(3), 297–328.

P. Baxendale 1958. *Machine-made index for technical literature - an experiment*, IBM Journal of Research Development, 2(4):354–361.

R. Bellman 1958. *On a routing problem*, Quarterly of Applied Mathematics 16:87–90.

F. Boudin and E. Morin. 2013 *Keyphrase Extraction for N-best reranking in multi-sentence compression*, Proceedigs of NAACL-HLT

G. Bouma. 2009. *Normalized (pointwise) Mutual information in collocation extraction*, Proceedings of GSCL.

J. Clarke and M. Lapata. 2008. *Global inference for sentence compression: An integer linear programming approach*, Journal of Artificial Intelligence Research, 31:399–429.

S. H. Corston-Oliver 2001. *Text compaction for display on very small screens*, Proceedings of the NAACL Workshop on Automatic Summarization.

E. W. Dijkstra 1959. *A note on two problems in connexion with graphs*, Numerische Mathematik 1:269–271.

B. Dorr, D. Zajic, and R. Schwartz. 2003. *Hedge trimmer: A parse-and-trim approach to headline generation*, Proceedings of the HLT-NAACL Workshop on Text Summarization.

H. P. Edmundson 1969. *New methods in automatic extracting*, Journal of the ACM, 16(2):264–285.

C. Fellbaum (Ed.). 1998. *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press.

K. Filippova. 2010. *Multi-sentence compression: Finding shortest paths in word graphs*, Proceedings of COLING.

K. Filippova and M. Strube. 2008. *Dependency tree based sentence compression*, Proceedings of INLG.

J. Ford, R. Lester 1956. *Network Flow Theory*, Paper P-923, Santa Monica, California: RAND Corporation.

M. Galley and K. R. McKeown. 2007. *Lexicalized Markov grammars for sentence compression*, Proceedings of NAACL-HLT

U. Hermjakob, A. Echihabi, and D. Marcu. 2002. *Natural language based reformulation resource and wide exploitation for question answering*, Proceedings of the Text Retrieval Conference.

C. Hori, S. Furui, R. Malkin, H. Yu, and A. Waibel. 2003. *A statistical approach to automatic speech summarization*, EURASIP Journal on Applied Signal Processing, 2:128–139.

H. Jing 2000. *Sentence reduction for automatic text summarization*, Proc. of ANLP.

H. Jing and K. McKeown. 2000. *Cut and paste based text summarization*, Proc. of NAACL.

N. Kaji, M. Okamoto, and S. Kurohashi,. 2004. *Paraphrasing predicates from written language to spoken language using the web*, Proceedings of HLT-NAACL.

J. Kupiec, J. Pedersen, and F. Chen 1995 *A trainable document summarizer*, Proceedings of SIGIR.

C. Lin. 2003. *Improving summarization performance by sentence compression - a pilot study*, Proceedings of the Int. Workshop on Information Retrieval with Asian Language.

C. Lin. 2004. *Rouge: A package for automatic evaluation of summaries*, Proceedings of the ACL Workshop on Text Summarization Branches Out.

C.-Y. Lin and E. H. Hovy. 2003. *Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics*, Proceedings of HLT-NAACL.

H.P. Luhn. 1958. *The Automatic Creation of Literature Abstracts*, IBM Journal of Research and Development 2(2), 159–165.

I. Mani. 2001. *Automatic Summarization*, Amsterdam, Philadelphia: John Benjamins.

D. Marcu. 1997 *The Rhetorical Parsing of Natural Language Texts*, Proceedings of ACL/EACL.

K. R. McKeown, J. Hirschberg, M. Galley, and S. Maskey. 2005. *From Text to Speech Summarization*, Proceedings of ICASSP.

G. A. Miller. 1995. *WordNet: a lexical database for English*, Communications of the ACM Vol. 38, No. 11: 39-41.

A. Nenkova, L. Vanderwende, and K. McKeown. 2006. *A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization*, Proceedings of SIGIR.

K. Owczarzak, J. M. Conroy, H. T. Dang, and A. Nenkova. 2012. *An assessment of the accuracy of automatic evaluation in summarization*, Proceedings of the Workshop on Evaluation Metrics and System Comparison for Automatic Summarization.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation*, Proceedings of ACL.

D. Roussinov and H. Chen. 2001 *Information Navigation on the Web by Clustering and Summarizing Query Results*, Information Processing and Management, 37 (6), 789–816.

G. Salton, J. Allan, C. Buckley, and A. Singhal, 1994 *Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts*, Science 264(5164), 1421–1426.

B. Taskar and D. Klein and M. Collins and D. Koller and C. Manning. 2004. *Max-margin parsing*, Proceedings of EMNLP, 2004.

S. Teufel and M. Moens. 1997 *sentence extraction as a classification task*, Proceedings of the ACL/EACL Workshop on Intelligent and Scalable Text Summarization.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. *Large margin methods for structured and interdependent output variables*, JMLR, 6 (Sep):1453-1484.

P. D. Turney. 2000. *Learning algorithms for keyphrase extraction*, Information Retrieval 2(4), 303–336.

S. Wan, R. Dale, M. Dras, C. Paris. 2007. *Global revision in summarisation : generating novel sentences with Prim's algorithm*, Proceedings of PACLING.

J. Y. Yen. 1971. *Finding the k shortest loopless paths in a network*, Management Science 17 (11): 712–716

M. Zinkevich, M. Weimer, A. Smola, and L. Li. 2011. *Parallelized stochastic gradient descent*, Proceedings of NIPS.