Learning to Fuse Disparate Sentences

Micha Elsner
School of Informatics
University of Edinburgh
melsner0@gmail.com

Deepak Santhanam
Brown Lab for
Linguistic Information Processing (BLLIP)
Department of Computer Science
Brown University, Providence, RI 02912
dsanthan@microsoft.com

Abstract

We present a system for fusing sentences which are drawn from the same source document but have different content. Unlike previous work, our approach is supervised, training on real-world examples of sentences fused by professional journalists in the process of editing news articles. Like Filippova and Strube (2008), our system merges dependency graphs using Integer Linear Programming. However, instead of aligning the inputs as a preprocess, we integrate the tasks of finding an alignment and selecting a merged sentence into a joint optimization problem, and learn parameters for this optimization using a structured online algorithm. Evaluation by human judges shows that our technique produces fused sentences that are both informative and readable.

1 Introduction

Sentence fusion is the process by which content from two or more original sentences is transformed into a single output sentence. It is usually studied in the context of multidocument summarization, since fusing similar sentences can avoid repetition of material which is shared by more than one input. However, human editors and summarizers do not restrict themselves to combining sentences which share most of their content. This paper extends previous work on fusion to the case in which the input sentences are drawn from the same document and express fundamentally different content, while still remaining related enough to make fusion sensible[1].

Our data comes from a corpus of news articles for which we have un-edited and edited versions. We search this corpus for sentences which were fused (or separated) by the editor; these constitute naturally occurring data for our system. One example from our dataset consists of input sentences (1) and (2) and output (3). We show corresponding regions of the input and output in boldface.

(1) **The bodies showed signs of torture**.

(2) They **were left on the side of a highway in Chilpancingo, about an hour north of the tourist resort of Acapulco** in the southern state of Guerrero, **state police** said.

(3) **The bodies** of the men, which **showed signs of torture**, **were left on the side of a highway in Chilpancingo, which is about an hour north of the tourist resort of Acapulco**, **state police** told Reuters.

While the two original sentences are linked by a common topic and reference to a shared entity, they are not paraphrases of one another. This could create a problem for traditional fusion systems which first find an alignment between similar dependency graphs, then extract a shared structure. While our system has the same basic framework of alignment and extraction, it performs the two jointly, as parts of a global optimization task. This makes it robust to uncertainty about the hidden correspondences between the sentences. We use structured online learning to find parameters for the system, allowing it to

[1] Unfortunately, we cannot release our corpus due to licensing agreements. Our system is available at bitbucket.org/melsner/sentencefusion.

Workshop on Monolingual Text-To-Text Generation, pages 54–63,

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 54–63,
Portland, Oregon, 24 June 2011.

discover good ways to piece together input sentences by examining examples from our corpus.

Sentence fusion is a common strategy in human-authored summaries of single documents– 36% of sentences in the summaries investigated by Jing and McKeown (1999) contain content from multiple sentences in the original document. This suggests that a method to fuse dissimilar sentences could be useful for single-document summarization. Our dataset is evidence that editing also involves fusing sentences, and thus that models of this task could contribute to systems for automatic editing.

In the remainder of the paper, we first give an overview of related work (Section 2). We next describe our dataset and preprocessing in more detail (Section 3), describe the optimization we perform (Section 4), and explain how we learn parameters for it (Section 5). Finally, we discuss our experimental evaluation and give results (Section 6).

2 Related work

Previous work on sentence fusion examines the task in the context of multidocument summarization, targeting groups of sentences with mostly redundant content. The pioneering work on fusion is Barzilay and McKeown (2005), which introduces the framework used by subsequent projects: they represent the inputs by dependency trees, align some words to merge the input trees into a lattice, and then extract a single, connected dependency tree as the output.

Our work most closely follows Filippova and Strube (2008), which proposes using Integer Linear Programming (ILP) for extraction of an output dependency tree. ILP allows specification of grammaticality constraints in terms of dependency relationships (Clarke and Lapata, 2008), as opposed to previous fusion methods (Barzilay and McKeown, 2005; Marsi and Krahmer, 2005) which used language modeling to extract their output.

In their ILP, Filippova and Strube (2008) optimize a function based on syntactic importance scores learned from a corpus of general text. While similar methods have been used for the related task of sentence compression, improvements can be obtained using supervised learning (Knight and Marcu, 2000; Turner and Charniak, 2005; Cohn and Lapata, 2009) if a suitable corpus of compressed sentences can be obtained. This paper is the first we know of to adopt the supervised strategy for sentence fusion.

For supervised learning to be effective, it is necessary to find or produce example data. Previous work does produce some examples written by humans, though these are used during evaluation, not for learning (a large corpus of fusions (McKeown et al., 2010) was recently compiled as a first step toward a supervised fusion system). However, they elicit these examples by asking experimental subjects to fuse selected input sentences– the choice of which sentences to fuse is made by the system, not the subjects. In contrast, our dataset consists of sentences humans actually chose to fuse as part of a practical writing task. Moreover, our sentences have disparate content, while previous work focuses on sentences whose content mostly overlaps.

Input sentences with differing content present a challenge to the models used in previous work. All these models use deterministic node alignment heuristics to merge the input dependency graphs. Filippova and Strube (2008) align all content words with the same lemma and part of speech; Barzilay and McKeown (2005) and Marsi and Krahmer (2005) use syntactic methods based on tree similarity. Neither method is likely to work well for our data. Lexical methods over-align, since there are many potential points of correspondence between our sentences, only some of which should be merged– "the Doha trade round" and "U.S. trade representative" share a word, but probably ought to remain separate regardless. Syntactic methods, on the other hand, are unlikely to find any alignments since the input sentences are not paraphrases and have very different trees. Our system selects the set of nodes to merge during ILP optimization, allowing it to choose correspondences that lead to a sensible overall solution.

3 Data and preprocessing

Our sentence fusion examples are drawn from a corpus of 516 pre- and post-editing articles from the Thomson-Reuters newswire, collected over a period of three months in 2008. We use a simple greedy method based on bigram count overlaps to align the sentences of each original article to sentences in the edited version, allowing us to find fused sentences.

Since these sentences are relatively rare, we use both merges (where the editor fused two input sentences) and splits (where the editor splits an input sentence into multiple outputs) as examples for our system. In the case of a split, we take the edited sentences as input for our method and attempt to produce the original through fusion[2]. This is suboptimal, since the editor's decision to split the sentences probably means the fused version is too long, but is required in this small dataset to avoid sparsity.

Out of a total of 9007 sentences in the corpus, our bigram method finds that 175 were split and 132 were merged, for a total of 307. We take 92 examples for testing and 189 for training[3].

Following previous work (Barzilay and McKeown, 2005), we adopt a labeled dependency format for our system's input. To produce this, we segment sentences with MXTerminator (Reynar and Ratnaparkhi, 1997) and parse the corpus with the self-trained Charniak parser (McClosky et al., 2006). We then convert to dependencies and apply rules to simplify and label the graph. An example dependency graph is shown in Figure 1.

We augment the dependency tree by adding a potential dependency labeled "relative clause" between each subject and its verb. This allows our system to transform main clauses, like "the bodies showed signs of torture", into NPs like "the bodies, which showed signs of torture", a common paraphrase strategy in our dataset.

We also add correspondences between the two sentences to the graph, marking nodes which the system might decide to merge while fusing the two sentences. We introduce correspondence arcs between pairs of probable synonyms[4]. We also annotate pronoun coreference by hand and create a correspondence between each pronoun and the heads of all coreferent NPs. The example sentence has only a single correspondence arc ("they" and "bodies") because input sentence (1) is extremely short, but most sentences have more.

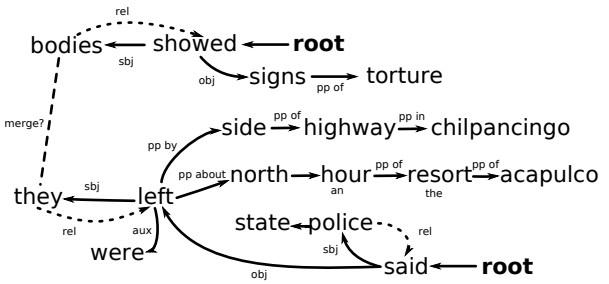

Figure 1: The labeled dependency graph for sentences (1) and (2). Dashed lines show a correspondence arc ("bodies" and "they") and potential relative clauses between subjects and VPs.

3.1 Retained information

Sentence fusion can be thought of as a two-part process: first, the editor decides which information from the input sentences to retain, and then they generate a sentence incorporating it. In this paper, we focus on the generation stage. To avoid having to perform content selection[5], we provide our system with the true information selected by the editor. To do this, we align the input sentences with the output by repeatedly finding the longest common substring (LCS) until a substring containing a matching content word can no longer be found. (The LCS is computed by a dynamic program similar to that for edit distance, but unlike edit distance, repeated LCS can handle reordering.) We provide our system with the boundaries of the retained regions as part of the input. For the example above, these are the regions of sentences (1) and (2) marked in boldface. Although this helps the system select the correct information, generating a grammatical and easy-to-read fused sentence is still non-trivial (see examples in section 7).

4 Fusion via optimization

Like Filippova and Strube (2008), we model our fusion task as a constrained optimization problem, which we solve using Integer Linear Programming (ILP). For each dependency from word w to head

[2]In a few cases, this creates two examples which share a sentence, since the editor sometimes splits content off from one sentence and merges it into another.

[3]We originally had 100 testing and 207 training examples, but found 26 of our examples were spurious, caused by faulty sentence segmentation.

[4]Words with the same part of speech whose similarity is greater than 3.0 according to the information-theoretic WordNet based similarity measure of Resnik (1995), using the implementation of (Pedersen et al., 2004).

[5]As pointed out by Daume III and Marcu (2004) and Krahmer et al. (2008), content selection is not only difficult, but also somewhat ill-defined without discourse context information.

h in the input sentences, we have a binary variable $x_{h,w}$, which is 1 if the dependency is retained in the output and 0 otherwise. However, unlike Filippova and Strube (2008), we do not know the points of correspondence between the inputs, only a set of possible points. Therefore, we also introduce 0-1 integer variables $m_{s,t}$ for each correspondence arc, which indicate whether word s in one sentence should be merged with word t in another. If the words are merged, they form a link between the two sentences, and only one of the pair appears in the output.

Each dependency x, each word w, and each merger m have an associated weight value v, which is assigned based on its features and the learned parameters of our system (explained in Section 5). Our objective function (4) sums these weight values for the structures we retain:

$$max \sum_{h,w} v_{h,w} \cdot v_w \cdot x_{h,w} + \sum_{s,t} v_{s,t} \cdot m_{s,t} \quad (4)$$

We use structural constraints to require the output to form a single connected tree. (In the following equations, W denotes the set of words, X denotes the set of dependencies and M denotes the potential correspondence pairs.) Constraint (5) requires a word to have at most one parent and (6) allows it to be merged with at most one other word. (7) and (8) require each merged node to have a single parent:

$$\forall w \in W, \sum_h x_{h,w} \leq 1 \quad (5)$$

$$\forall w \in W, \sum_t m_{s,t} \leq 1 \quad (6)$$

$$\forall s,t \in M, m_{s,t} \leq \sum_h x_{h,s} + \sum_h x_{h,t} \quad (7)$$

$$\forall s,t \in M, m_{s,t} + \sum_h x_{h,s} + \sum_h x_{h,t} \leq 2 \quad (8)$$

(9) forces the output to be connected by ensuring that if a node has children, it either has a parent or is merged.

$$\forall_w \in W, \sum_c x_{c,w} - |W| \sum_h x_{h,w} - |W| \sum_u m_{u,w} \leq 0 \quad (9)$$

Certain choices of nodes to merge or dependencies to follow can create a cycle, so we also introduce a rank variable $r_w \in \mathbb{R}$ for each word and constrain each word (except the root) to have a higher rank than its parent (10). Merged nodes must have equal ranks (11).

$$\forall_{w,h} \in X, |X| x_{h,w} + r_h - r_w \leq |X| - 1 \quad (10)$$

$$\forall_{s,t} \in M, |X| m_{s,t} + r_s - r_t \leq |X| \quad (11)$$

We also apply syntactic constraints to make sure we supply all the required arguments for each word we select. We hand-write rules to prevent the system from pruning determiners, auxiliary verbs, subjects, objects, verbal particles and the word "not" unless their head word is also pruned or it can find a replacement argument of the same type. We learn probabilities for prepositional and subclause arguments using the estimation method described in Filippova and Strube (2008), which counts how often the argument appears with the head word in a large corpus. While they use these probabilities in the objective function, we threshold them and supply constraints to make sure all argument types with probability $> 10\%$ appear if the head is chosen.

Word merging makes it more difficult to write constraints for required arguments, because a word s might be merged with some other word t which is attached to the correct argument type (for instance, if s and t are both verbs and they are merged, only one of them must be attached to a subject). This condition is modeled by the expression $m_{s,t} \cdot x_{t,a}$, where a is a argument word of the appropriate type. This expression is non-linear and cannot appear directly in a constraint, but we can introduce an auxiliary variable $g_{s,t,A}$ which summarizes it for a set of potential arguments A, while retaining a polynomial-sized program:

$$\forall_{s,t} \in M, \sum_{a \in A} x_{a,s} + \sum_{a \in A} x_{a,t} + |W| m_{s,t} - |W+1| g_{s,t,A} \geq 0 \quad (12)$$

(13) then requires a word s to be connected to an argument in set A, either via a link or directly:

$$\sum_h x_{s,h} - 2 \sum_{t:\{s,t \in M\}} g_{s,t,A} - 2 \sum_{a \in A} x_{a,s} \leq 0 \quad (13)$$

The resulting resulting ILP is usually solvable within a second using CPLEX (Ilog, Inc., 2003).

4.1 Linearization

The output of the ILP is a dependency tree, not an ordered sentence. We determine the final ordering mostly according to the original word order of the input. In the case of a merged node, however, we must also interleave modifiers of the merged heads, which are not ordered with respect to one another. We use a simple heuristic, trying to place dependencies with the same arc label next to one another; this can cause errors. We must also introduce conjunctions between arguments of the same syntactic type; our system always inserts "and". Finally, we choose a realization for the dummy relative pronoun *THAT* using a trigram language model (Stolcke, 2002). A more sophisticated approach (Filippova and Strube, 2009) might lead to better results.

5 Learning

The solution which the system finds depends on the weights v which we provide for each dependency, word and merger. We set the weights based on a dot product of features ϕ and parameters α, which we learn from data using a supervised structured technique (Collins, 2002). To do so, we define a loss function $L(s, s') \rightarrow R$ which measures how poor solution s is when the true solution is s'. For each of our training examples, we compute the *oracle* solution, the best solution accessible to our system, by minimizing the loss. Finally, we use the structured averaged perceptron update rule to push our system's parameters away from bad solutions and towards the oracle solutions for each example.

Our loss function is designed to measure the high-level similarity between two dependency trees containing some aligned regions. (For our system, these are the regions found by LCS alignment of the input strings with the output.) For two sentences to be similar, they should have similar links between the regions. Specifically, we define the *paths* $P(s, C)$ in a tree s with a set of regions C as the set of word pairs w, w' where w is in one region, w' is in another, and the dependency path between w and w' lies entirely outside all the regions. An example is given in figure 2.

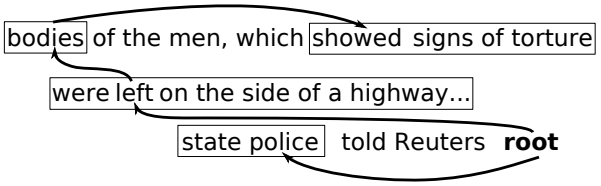

Figure 2: Paths between retained regions in sentence (3). Boxes indicate the retained regions.

Our loss (equation 14) is defined as the number of paths in s and s' which do not match, plus a penalty K_1 for keeping extra words, minus a bonus K_2 for retaining words inside aligned regions:

$$\begin{aligned} &L(s,s';C,K) = \\ &\quad |(P(s,C) \cup P(s',C)) \setminus (P(s,C) \cap P(s',C))| \\ &\quad + K_1 |w \in s \setminus C| - K_2 |w \in s \cap C| \end{aligned} \quad (14)$$

To compute the oracle s^*, we must minimize this loss function with respect to the human-authored reference sentence r over the space S of fused dependency trees our system can produce.

$$s^* = argmin_{s \in S} \, L(s, r) \quad (15)$$

We perform the minimization by again using ILP, keeping the constraints from the original program but setting the objective to minimize the loss. This cannot be done directly, since the existence of a path from s to t must be modeled as a product of x variables for the dependencies forming the path. However, we can again introduce a polynomial number of auxiliary variables to solve the problem. We introduce a 0-1 variable $q^s_{h,w}$ for each path start word s and dependency h, w, indicating whether the dependency from h to w is retained and forms part of a path from s. Likewise, we create variables q^s_w for each word and $q^s_{u,v}$ for mergers[6]. Using these variables, we can state the loss function linearly as (16),

[6]The q variables are constrained to have the appropriate values in the same way as (12) constrains g. We will print the specific equations in a technical report when this work is published.

where $P(r, C)$ is the set of paths extracted from the reference solution.

$$min \sum_{s,t} q^s_{h,t} - 2 \sum_{s,t \in P(r,C)} q^s_{h,t} \quad (16)$$

The oracle fused sentence for the example (1) and (2) is (17). The reference has a path from *bodies* to *showed*, so the oracle includes one as well. To do so, follows a relative clause arc, which was not in the original dependency tree but was created as an alternative by our syntactic analysis. (At this stage of processing, we show the dummy relative pronoun as *THAT*.) It creates a path from *left* to *bodies* by choosing to merge the pronoun *they* with its antecedent. Other options, such as linking the two original sentences with "and", are penalized because they would create erroneous paths– since there is no direct path between *root* and *showed*, the oracle should not make *showed* the head of its own clause.

(17) the bodies THAT showed signs of torture were left on the side of a highway in Chilpancingo about an hour north of the tourist resort of Acapulco state police said

The features which represent each merger, word and dependency are listed in Table 1. We use the first letters of POS tags (in the Penn Treebank encoding) to capture coarse groupings such as all nouns and all verbs. For mergers, we use two measures of semantic similarity, one based on Roget's Thesaurus (Jarmasz and Szpakowicz, 2003) and another based on WordNet (Resnik, 1995). As previously stated, we hand-annotate the corpus with true pronoun coreference relationships (about 30% of sentences contain a coreferent pronoun). Finally, we provide the LCS retained region boundaries as explained above.

Once we have defined the feature representation and the loss function, and can calculate the oracle for each datapoint, we can easily apply any structured online learning algorithm to optimize the parameters. We adopt the averaged perceptron, applied to structured learning by (Collins, 2002). For each example, we extract a current solution s_t by solving the ILP (with weights v dependent on our parameters α), then perform an update to α which forces the system away from s_t and towards the oracle solution s^*. The update at each timestep t (18) depends on the loss, the global feature vectors Φ, and a learning rate parameter η. (Note that the update leaves the parameters unchanged if the loss relative to the oracle is 0, or if the two solutions cannot be distinguished in terms of their feature vectors.)

COMPONENT	FEATURES
MERGER	SAME WORD SAME POS TAGS SAME FIRST LETTER OF THE POS TAGS POS TAG IF WORD IS SAME COREFERENT PRONOUN SAME DEPENDENCY ARC LABEL TO PARENT ROGET'S SIMILARITY WORDNET SIMILARITY FIRST LETTER OF BOTH POS TAGS
WORD	POS TAG AND ITS FIRST LETTER WORD IS PART OF RETAINED CHUNK IN EDITOR'S FUSION
DEPENDENCY	POS TAGS OF THE PARENT AND CHILD FIRST LETTER OF THE POS TAGS TYPE OF THE DEPENDENCY DEPENDENCY IS AN INSERTED RELATIVE CLAUSE ARC PARENT IS RETAINED IN EDITOR'S SENTENCE CHILD IS RETAINED IN EDITOR'S SENTENCE

Table 1: List of Features.

$$\alpha_{t+1} = \alpha_t + \eta(L(s_t, r) - L(s^*, r))(\Phi(s^*) - \Phi(s_t)) \quad (18)$$

We do 100 passes over the training data, with η decaying exponentially toward 0. At the end of each pass over the data, we set $\hat{\alpha}$ to the average of all the α_t for that pass (Freund and Schapire, 1999). Finally, at the end of training, we select the committee of 10 $\hat{\alpha}$ which achieved lowest overall loss and average them to derive our final weights (Elsas et al., 2008). Since the loss function is nonsmooth, loss does not decrease on every pass, but it declines overall as the algorithm proceeds.

6 Evaluation

Evaluating sentence fusion is a notoriously difficult task (Filippova and Strube, 2008; Daume III and Marcu, 2004) with no accepted quantitative metrics, so we have to depend on human judges for evaluation. We compare sentences produced by our system to three alternatives: the editor's fused sentence, a readability upper-bound and a baseline formed by splicing the input sentences together by inserting the word "and" between each one. The readability upper

bound is the output of parsing and linearization on the editor's original sentence (Filippova and Strube, 2008); it is designed to measure the loss in grammaticality due to our preprocessing.

Native English speakers rated the fused sentences with respect to readability and content on a scale of 1 to 5 (we give a scoring rubric based on (Nomoto, 2009)). 12 judges participated in the study, for a total of 1062 evaluations[7]. Each judge saw the each pair of inputs with the retained regions boldfaced, plus a single fusion drawn randomly from among the four systems. Results are displayed in Table 2.

System	Readability	Content
Editor	4.55	4.56
Readability UB	3.97	4.27
"And"-splice	3.65	3.80
Our System	3.12	3.83

Table 2: Results of human evaluation.

7 Discussion

Readability scores indicate that the judges prefer human-authored sentences, then the readability upper bound, then "and"-splicing and finally our system. This ordering is unsuprising considering that our system is abstractive and can make grammatical errors, while the remaining systems are all based on grammatical human-authored text. The gap of .58 between human sentences and the readability upper bound represents loss due to poor linearization; this accounts for over half the gap between our system and human performance.

For content, the human-authored sentences slightly outperform the readability upper bound– this indicates that poor linearization has some effect on content as well as readability. Our system is slightly better than "and"-splicing. The distribution of scores is shown in Table 3. The system gets more scores of 5 (perfect), but it occasionally fails drastically and receives a very low score; "and"-splicing shows less variance.

Both metrics show that, while our system does not achieve human performance, it does not lag behind by that much. It performs quite well on some relatively hard sentences and gets easy fusions right most of the time. For instance, the output on our example sentence is (19), matching the oracle (17).

[7]One judge completed only the first 50 evaluations; the rest did all 92.

	1	2	3	4	5	Total
"And"-splice	3	43	60	57	103	266
System	24	24	39	58	115	260

Table 3: Number of times each **Content** score was assigned by human judges.

(19) The bodies who showed signs of torture were left on the side of a highway in Chilpancingo about an hour north of the tourist resort of Acapulco state police said.

In some cases, the system output corresponds to the "and"-splice baseline, but in many cases, the "and"-splice baseline adds extraneous content. While the average length of a human-authored fusion is 34 words, the average splice is 49 words long. Plainly, editors often prefer to produce compact fusions rather than splices. Our own system's output has an average length of 33 words per sentence, showing that it has properly learned to trim away extraneous information from the input. We instructed participants to penalize the content score when fused sentences lost important information or added extra details.

Our integration of node alignment into our solution procedure helps the system to find good correspondences between the inputs. For inputs (20) and (21), the system was allowed to match "company" to "unit", but could also match "terrorism" to "administration" or to "lawsuit". Our system correctly merges "company" and "unit", but not the other two pairs, to form our output (22); the editor makes the same decision in their fused sentence (23).

(20) The suit **claims** the company **helped fly terrorism suspects abroad to secret prisons**.

(21) Holder's **review was disclosed the same day as Justice Department lawyers repeated a Bush administration** state-secret **claim in a lawsuit against a Boeing Co unit**.

(22) Review was disclosed the same day as Justice Department lawyers repeated a Bush administration claim in a lawsuit against a Boeing Co unit that helped fly terrorism suspects abroad to secret prisons.

(23) The review was disclosed the same day that Justice Department lawyers repeated Bush administration claims of state secrets in a lawsuit against a Boeing Co <BA.N> unit claiming it helped fly terrorism suspects abroad to secret prisons.

In many cases, even when the result is awkward or ungrammatical, the ILP system makes reasonable choices of mergers and dependencies to retain. For inputs (24) and (25), the system (26) decides "Secretary-General" belongs as a modifier on "de Mello", which is in fact the choice made by the editor (27). In order to add the relative clause, the editor paraphrased "de Mello's death" as "de Mello was killed". Our system, without this paraphrase option, is forced to produce the improper phrase "de Mello's death who"; a wider array of paraphrase options might lead to better results.

This example also demonstrates that the system does not simply keep the LCS-aligned retained regions and throw away everything else, since the result would be ungrammatical. Here it links the selected content by also choosing to keep "could have been", "an account" and "death".

(24) **Barker** mixes an account of **Vieira de Mello**'s death with scenes from his **career, which included working in countries such as Mozambique, Cyprus, Cambodia, Bangladesh, and the former Yugoslavia**.

(25) Had he lived, he could have been **a future U.N. Secretary-General.**

(26) Barker mixes an account of Vieira de Mello's death who could been a future U.N. secretary-general with scenes from career which included working in countries as such Mozambique Cyprus Cambodia and Bangladesh

(27) Barker recounted the day Vieira de Mello, a Brazilian who was widely tipped as a future U.N. Secretary-General, was killed and mixes in the story of the 55-year-old's career, which included working in countries such as Mozambique, Cyprus, Cambodia, Bangladesh, and Yugoslavia.

Many of our errors are due to our simplistic linearization. For instance, we produce a sentence beginning "Biden a veteran Democratic senator from Delaware that Vice president-elect and Joe...", where a correct linearization of the output tree would have begun "Vice President-elect Joe Biden, a veteran Democratic senator from Delaware that...". Some errors also occur during the ILP tree extraction process. In (28), the system fails to mark the arguments of "took" and "position" as required, leading to their omission, which makes the output ungrammatical.

(28) The White House that took when Israel invaded Lebanon in 2006 showed no signs of preparing to call for restraint by Israel and the stance echoed of the position.

8 Conclusion

We present a supervised method for learning to fuse disparate sentences. To the best of our knowledge, it is the first attempt at supervised learning for this task. We apply our method to naturally occurring sentences from editing data. Despite using text generation, our system is comparable to a non-abstractive baseline.

Our technique is general enough to apply to conventional fusion of similar sentences as well– all that is needed is a suitable training dataset. We hope to make use of the new corpus of McKeown et al. (2010) for this purpose. We are also interested in evaluating our approach on the fused sentences in abstractive single-document summaries.

The performance of our readability upper bound suggests we could improve our results using better tree linearization techniques and parsing. Although we show results for our system using hand-annotated pronoun coreference, it should be possible to use automatic coreference resolution instead.

Paraphrase rules would help our system replicate some output structures it is currently unable to match (for instance, it cannot convert between the copular "X is Y" and appositive "X, a Y" constructions). Currently, the system has just one such

rule, which converts main clauses to relatives. Others could potentially be learned from a corpus, as in (Cohn and Lapata, 2009).

Finally, in this study, we deliberately avoid investigating the way editors choose which sentences to fuse and what content from each of them to retain. This is a challenging discourse problem that deserves further study.

Acknowledgements

We are very grateful to Alan Elsner, Howard Goller and Thomas Kim at Thomson-Reuters for giving us access to this dataset. We thank Eugene Charniak for his supervision, our colleagues in BLLIP for their comments, Kapil Thadani and Kathy McKeown for discussing the project with us, and our human evaluators for completing a task which turned out to be extremely tedious. Part of this work was funded by a Google Fellowship in Natural Language Processing.

References

Regina Barzilay and Kathleen McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res. (JAIR)*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *J. Artif. Intell. Res. (JAIR)*, 34:637–674.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Hal Daume III and Daniel Marcu. 2004. Generic sentence fusion is an ill-defined summarization task. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 96–103, Barcelona, Spain, July. Association for Computational Linguistics.

Jonathan L. Elsas, Vitor R. Carvalho, and Jaime G. Carbonell. 2008. Fast learning of document ranking functions with the committee perceptron. In *WSDM*, pages 55–64.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185, Honolulu, Hawaii, October. Association for Computational Linguistics.

Katja Filippova and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 225–228, Boulder, Colorado, June. Association for Computational Linguistics.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Ilog, Inc. 2003. Cplex solver.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's thesaurus and semantic similarity. In *Conference on Recent Advances in Natural Language Processing*, pages 212–219.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *SIGIR*, pages 129–136.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 703–71.

Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of ACL-08: HLT, Short Papers*, pages 193–196, Columbus, Ohio, June. Association for Computational Linguistics.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the 10th European Workshop on Natural Language Generation*, pages 109–117.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320, Los Angeles, California, June. Association for Computational Linguistics.

Tadashi Nomoto. 2009. A comparison of model free versus model intensive approaches to sentence compression. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 391–399, Singapore, August. Association for Computational Linguistics.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::Similarity - measuring the relatedness of concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington D.C.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 257–286, November.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proc. Assoc. for Computational Linguistics (ACL)*, pages 290–297.