Learning To Fuse Disparate Sentences

By

Deepak Santhanam

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Department of Computer Science at

Brown University

PROVIDENCE, RHODE ISLAND

DECEMBER 2010

Abstract

The work presented here is based on a paper under submission with Micha Elsner.

This thesis presents a system for fusing sentences which are drawn from the same souce document but differ significantly in content. The system described here is supervised, trained from real world examples of sentences fused by professional journalists in the process of editing news articles. The system is similar to previous work in that, it merges dependency graphs using Integer Linear Programming. Instead of aligning the inputs as a preprocess, it integrates the task of finding an alignment along with the selecting a merged sentence into a single joint optimization problem for which the parameters are learned using a structured online learning algorithm. Human evaluation shows that the technique produces fused sentences which are both informative and highly readable.

Acknowledgements

I wish to thank Eugene Charniak for being a great advisor and teacher. Working with him has been a wonderful experience. I must thank Micha Elsner, who has been a great friend and mentor. This thesis is based on work I did with him in which he contributed significantly and also helped me out whenever I got stuck. For the past year and half, Eugene, him and everyone else in BLLIP have patiently put up with me and my questions and were always there to help me out whenever i needed them.

I wish to thank Michael Black for his amazing vision courses and also for giving me an opportunity to work with him for a semester. I also thank Erik Sudderth for his awesome machine learning courses both of which I thoroughly enjoyed.

I thank all my friends in the department, tstaff and astaff for making my life easier during my time here.

Finally, I thank my parents and grandparents for their unconditional love and funding.

Contents

Chapter 1

Introduction

1.1 Sentence Fusion

Text summarization is a useful technique and a popular research topic in computational linguistics. Most summarization systems (single and multi document) today use extractive methods for creating their summaries. In the single document case, they just extract whole sentences from the source document and produce an ordered set of these sentences as a summary. In the multidocument case, the summary is constructed by extracting sentences spanning across multiple documents and ordering them. It could be the case that sentences extracted from different documents might have content overlap and the summaries produced by these systems may contain a significant amount of textual redundancy[7] which needs to be reduced.

Sentence fusion [1, 7] is a text-to-text generation scheme where two or more sentences are "fused" to form one sentence which contains combined information from the input sentences. This technique is usually studied in the context of multi-document summarization as fusing sentences which convey similar information may reduce textual redundancy in summaries. Traditional fusion systems have looked at the case where the sentences to be fused express the same content while exhibiting different sentence structure and words. In this work, we explore the case where the input sentences do not share most of their content, but are related to the extent that fusing them together will be sensible. Usually, such sentences are linked by a common topic or refer to a shared entity and will not be a paraphrase of each other as humans do not restrict themselves to fuse sentences which share most of their content.

Jing and McKeown[12] state that in their investigation of summaries, 36% of the sentences contain content from multiple sentences present in the original document. It can be postulated that a document does not contain a lot of redundant sentences and it is reasonable to suggest that these sentences were formed by fusing disparate content. This technique could prove to be very useful for abstractive summarization where instead of taking whole sentences and re-ordering them, summaries are produced by combining information across multiple source sentences.

The utility of sentence fusion might not be limited to just abstractive summarization. An interesting application would be in machine translation where suitable sentences in the source document could be fused and then translated to the target language to produce a summarized translation which could possibly be better than translating sentence by sentence. Also, the way in which we extracted our dataset presents another application. Professional human editors perform sentence fusion as part of the document editing task. This technique could be used as part of an automatic document editing system.

1.2 Previous work

The pioneering work on fusion is by Barzilay and McKeown [1] who introduced the basic framework on which subsequent work was done. They used sentence fusion for multidocument summarization where sentences with redundant content were chosen to be fused. The input sentences were represented using dependency trees and then word alignments were found. Then they merge some aligned words to form a fusion lattice and finally, extract a single connected dependency tree as the output sentence.

Filippova and Strube[7] used Integer Linear Programming to extract a dependency tree for the output sentence. Prior work[1, 14] had used language modeling for extracting the output tree. But this work used Integer Linear Programming and their ILP had options to impose gramaticality constraints in terms of dependency relationships [2] and it also attached weights for each dependency arc which represented syntactic importance scores learned from a parsed corpus. The score on the arcs predict whether a dependent is a required argument or an optional one. They finally use a complex linearizer to form the actual output sentence. Their work was done for German.

1.3 A Supervised Joint Approach

Our system has the same basic framework as previous ones i.e. it performs alignment and extraction. Unlike previous work, we merge these two steps into one single joint optimization step. The rationale behind this approach is that it will make the system robust to uncertainity about any hidden corresspondences between the sentences. Also, our system is capable of fusing disparate sentences which as mentioned before will not be paraphrases of each other and this type of input might pose difficulties to existing sentence fusion systems. We use structured online learning to learn the parameters for the joint global optimization task allowing it to learn good ways to combine input sentences from our training corpus.

Input sentences with disparate content present a challenge to existing systems as all these models use deterministic node alignment using heuristics and merge the input dependency graphs. [7] align all content words with the same lemma and parts of speech. [1, 14] use syntactic methods based on tree similarity. Lexical methods might over align in case of disparate sentences as there are many points of correspondence between disparate sentences and only some of these correspondences should actually be merged. Syntactic methods are unlikely to find alignments since the input sentences are not paraphrases and form different trees. Since our model selects the set of nodes to merge during the joint optimization step, it can choose correspondences which lead to a more sensible global solution.

Chapter 2

Fusing Disparate Sentences

2.1 Training Data

Our training data comes from a corpus of edited and un-edited news articles where the editor either fused certain sentences during the edit-process or spliced a fused sentence. A typical training example would be,

(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.

There are 516 articles from Thompson-Reuters newswire, collected over a period of three months in 2008. Using a greedy method based on bi-gram counts, we find overlappings in the original and edited versions of the articles and form our dataset. Due to the scarcity of merged sentences, we also take sentences which were spliced, i.e. large sentences split into smaller ones during the edit process as training examples. However in this case, we take the edited sentences and try to produce the original fused sentence.

There were 9007 sentences in the corpus out of which, 175 splits and 132 merges were found. We randomly selected 100 from these and marked them as test data and kept the remainder as training. Due to faulty sentence segmentations, we later found that 26 of these examples were bad and we manualy removed them. The final test set had 92 examples and training set had 189 examples which we also used for development.

2.2 Preprocessing And Retained Information

We use a labeled dependency format for our system's input. First, the sentences are segmented using MXTerminator [20] and parsed using the self trained Charniak parser[15]. Then, using a heuristic procedure, we convert it into a dependency format and after that, produce a simplified labeled graph. An example of such a graph is shown in Fig 2.1.

We augment this dependency tree by adding a potential dependency labeled "relative clause " between every subject and verb as this allows our system to transform main clauses like "the bodies showed signs of torture" into noun phrases like "the bodies which showed signs of torture" as this was a common paraphrasing strategy

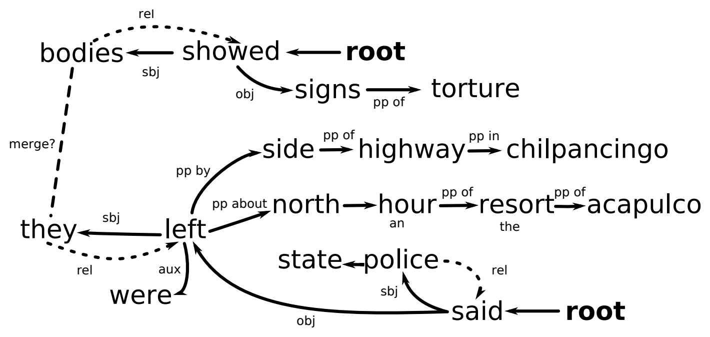

Figure 2.1: **The labeled dependency graphs for the example sentences. Dashed lines show a correspondence arc and potential relative clauses between subjects and verb phrases.**

in the dataset. Correspondences between the two sentences are added by marking nodes which the system might wish to merge during the joint optimization step. Correspondence arcs are introduced between all pairs of words with the same parts of speech who get a wordNet based similarity measure [19, 18] greater than 3.0. Pronoun co-reference is annotated by hand and we create a correspondence between each pronoun and all the heads of all coreferent noun phrases.

Fusion is a two-step process involving content selection and merging. Daume III and Marcu [5] point out that the choice of information selected to be retained is unpredictable using only sentence level information and Krahmer[13] says that it is easy if some form of discourse context is present, for example, a question to be answered.

This work primarily focusses on the problem of sentence-level generation as opposed to information selection and as a consequence, we provide our system with the true selection information, i.e. what the editor actually selected to merge. We do this by aligning the input sentences with the output by repeatedly finding the Longest common substring untill a substring containing matching content words can no longer be found. It is to be noted that LCS can handle reordering unlike edit distance. The boundaries of the retained regions are also provided as a input to our system. In the first example, the boldface parts indicate the content which was selected to be retained.

2.3 Fusion Through Joint Optimization

Following [7], we model the fusion task as a constrained optimization problem which we attempt to solve using ILP. For each dependency from word w to head 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. We are unaware of the points of correspondence between the input sentences and know of only a set of possible points. Hence, we also introduce 0-1 integer variables $m_{s,t}$ for each correspondence arc, indicating whether word s in one sentence should be merged with word t in another and if merged, whether they form a link between the two sentences, and finally determine if 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 the next section). Our objective function which sums these weight values for the structures we retain is given by

$$\mathbf{max} \sum_{\mathbf{h,w}} \mathbf{v_{h,w} \cdot v_w \cdot x_{h,w}} + \sum_{\mathbf{s,t}} \mathbf{v_{s,t} \cdot m_{s,t}} \qquad - \qquad (\mathbf{1})$$

Structural constrains are used to make the output form a single connected tree. The following constraint ensures that every word should have at most one parent

$$\forall \mathbf{w} \in \mathbf{W},\ \sum_{\mathbf{h}} \mathbf{x_{h,w}} \leq \mathbf{1} \qquad - \qquad (\mathbf{2})$$

The next constraint is to allow a word to be merged with at most one other word.

$$\forall \mathbf{s}, \mathbf{t} \in \mathbf{M},\ \mathbf{m_{s,t}} + \sum_{\mathbf{h}} \mathbf{x_{h,s}} + \sum_{\mathbf{h}} \mathbf{x_{h,t}} \leq \mathbf{2} \qquad - \qquad (\mathbf{3})$$

The following two constrains make sure that each merged node has one single parent.

$$\forall \mathbf{s}, \mathbf{t} \in \mathbf{M},\ \mathbf{m_{s,t}} \leq \sum_{\mathbf{h}} \mathbf{x_{h,s}} + \sum_{\mathbf{h}} \mathbf{x_{h,t}} \qquad - \qquad (\mathbf{4})$$

$$\forall \mathbf{s}, \mathbf{t} \in \mathbf{M},\ \mathbf{m_{s,t}} + \sum_{\mathbf{h}} \mathbf{x_{h,s}} + \sum_{\mathbf{h}} \mathbf{x_{h,t}} \leq \mathbf{2} \qquad - \qquad (\mathbf{5})$$

The following constraint forces the output to be connected by ensuring that if a node has children, it either has a parent or is merged.

$$\forall_{\mathbf{w}} \in \mathbf{W},\ \sum_{\mathbf{c}} \mathbf{x_{c,w}} - |\mathbf{W}| \sum_{\mathbf{h}} \mathbf{x_{h,w}} - |\mathbf{W}| \sum_{\mathbf{u}} \mathbf{m_{u,w}} \leq \mathbf{0} \qquad - \qquad (\mathbf{6})$$

When merging the nodes, it is possible that certain choices of the merges might lead to a cycle being formed. To avoid this, a rank variable $r_w \in \mathbb{R}$ for each word and make it have a higher rank than the word's parent. We do not do this for the root.

$$\forall_{\mathbf{w,h}} \in \mathbf{X}, |\mathbf{X}|\mathbf{x_{h,w}} + \mathbf{r_h} - \mathbf{r_w} \leq |\mathbf{X}| - \mathbf{1} \qquad - \qquad (\mathbf{7})$$

The next constraint is to make merged nodes have equal ranks

$$\forall_{\mathbf{s,t}} \in \mathbf{M}, |\mathbf{X}|\mathbf{m_{s,t}} + \mathbf{r_s} - \mathbf{r_t} \leq |\mathbf{X}| \qquad - \qquad (\mathbf{8})$$

It is necessary that required arguments are present for each word we select to be in the final sentence. This could be made possible by imposing syntactic constraints and rules are placed to prevent the system to not prune away determiners, auxiliary verbs, objects, subjects, verbal particles and the word “not” unless their head word is also pruned or a suitable replacement argument of the same type can be found by the solver. Following [7], we learn probabilities for prepositional arguments and subclass arguments. We estimate how often a particular argument appears with a particular word in a large corpus. While [7]use this in their objective function, we estimate the probabilities and then threshold and supply constraints to make sure all argument types with a probability greater than 10% appear if the corresponding head word is chosen.

It is difficult to come up with constraints for required arguments as word merging makes it very complex. A word s might be merged with another word t attached to the right type of argument. An example of this would be if two verbs are merged, only one of them should be attached to a subject. This is modeled by the expresssion, $\mathbf{m_{s,t} \cdot x_{t,a}}$ 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:

$$\forall_{\mathbf{s,t}} \in \mathbf{M},\ \sum_{\mathbf{a \in A}} \mathbf{x_{a,s}} + \sum_{\mathbf{a \in A}} \mathbf{x_{a,t}} + |\mathbf{W}|\mathbf{m_{s,t}} - |\mathbf{W+1}|\mathbf{g_{s,t,A}} \geq \mathbf{0} \qquad - \qquad (\mathbf{9})$$

The final constraint requires a word s to be connected to an argument in the set A either directly or through a link.

$$\sum_{\mathbf{h}} \mathbf{x_{s,h}} - \mathbf{2} \sum_{\mathbf{t:\{s,t \in M\}}} \mathbf{g_{s,t,A}} - \mathbf{2} \sum_{\mathbf{a \in A}} \mathbf{x_{a,s}} \leq \mathbf{0} \qquad - \qquad (\mathbf{10})$$

The ILP formed is solved using CPLEX [10] and it was observed that it usually takes less than a second to come up with a solution. In rare cases, it can take much longer than a second to solve this ILP, but we cut off the solver after 10 seconds and use the best available solution.

2.4 Online Learning Of Parameters

We have to provide weights v for each dependency, word and potential merger to the system in order for it to find a good solution. The weights are based on a dot product of features ϕ and parameters α,which we learn from data using a supervised structured technique [4]. 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.

The loss function is designed to measure the similarity between the two dependency trees which contain aligned regions of words and we find this for our system using LCS alignment of the input strings with the output. Our system retains the structire within each region and is incapable of replicating any structures which lie entirely outside the aligned regions. so we concentrate on paths between regions. Specifically, we define the *paths* $P(s, C)$ in a tree 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 Fig 2.2.

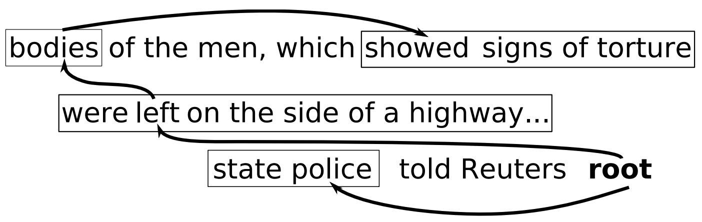

Figure 2.2: **Paths between retained regions in sentence. Boxes indicate retained regions.**

The following loss function is defined as the number of paths in s and s' which do not match and a penalty K_1 for keeping extra words and a bonus K_2 for retaining words inside aligned regions.

The loss function $\mathbf{L}(\mathbf{s}, \mathbf{s}'; \mathbf{C}, \mathbf{K})$ is given by

$$|(\mathbf{P}(\mathbf{s},\mathbf{C}) \cup \mathbf{P}(\mathbf{s}',\mathbf{C})) \ \setminus \ (\mathbf{P}(\mathbf{s},\mathbf{C}) \cap \mathbf{P}(\mathbf{s}',\mathbf{C}))| + \mathbf{K_1}|\mathbf{w} \in \mathbf{s} \setminus \mathbf{C}| - \mathbf{K_2}|\mathbf{w} \in \mathbf{s} \cap \mathbf{C}| - (\mathbf{11})$$

For computing the oracle $s*$, the above loss function is minimized with respect to the human authored reference sentence r over the space S of fused dependency trees the system can produce.

This optimization is again performed by using ILP keeping the same original constraints but modifying 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. Using these variables, we can state the loss function linearly as

$$\mathbf{min} \sum_{\mathbf{s,t}} \mathbf{q^s_{h,t}} - \mathbf{2} \sum_{\mathbf{s,t \in P(r,C)}} \mathbf{q^s_{h,t}} \qquad - \qquad \mathbf{(12)}$$

where $P(r, C)$ is the set of paths extracted from the reference solution. The oracle sentence for the example shown before in section 2.1 is

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 oracle creates a path from *bodies* to *torture* by following 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.

The features which represent each merger, word and dependency are listed in Table 2.1. 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[11] and another based on WordNet [19]. As previously stated, we also hand-annotate the corpus with true pronoun coreference relationships. Finally, we provide the retained regions found using LCS alignment between the inputs and the editor's output. This allows us to focus on the syntactic problem of generating a good fusion, rather than trying to

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 2.1: List of Features.

guess what content to select.

Once we have defined the feature representation and the loss function, we calculate the oracle for each datapoint and for this, we can easily apply any structured online learning algorithm to optimize the parameters. We adopt the averaged perceptron, applied to structured learning by [4] 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^*.

$$\alpha_{\mathbf{t+1}} = \alpha_{\mathbf{t}} + \eta(\mathbf{L}(\mathbf{s_t}, \mathbf{r}) - \mathbf{L}(\mathbf{s}^*, \mathbf{r}))(\boldsymbol{\Phi}(\mathbf{s}^*) - \boldsymbol{\Phi}(\mathbf{s_t})) \qquad - \qquad \mathbf{(13)}$$

The update at each timestep t (13) depends on the loss, the global feature vectors Φ, and a learning rate parameter η. 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.

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 [9]. 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 [6]. Since the loss function is nonsmooth, loss does not decrease on every pass, but it declines overall as the algorithm proceeds.

2.5 Tree Linearization

The result of the optimization step is a dependency tree which needs to be flattened to a sentence. The final ordering of the words depend on the original word order of the input sentences. For merged nodes, modifiers of the merged heads which are not ordered with respect to each other should be interleaved. The scheme we use involves trying to place dependencies with the same arc label next to one another.

Conjunctions (our system inserts only "and") should also be introduced between arguments of the same syntactic type. We also choose the dummy realization of the relative pronoun THAT using a trigram language model [21]. Since the focus of the work was not on linearization, but rather on the selection and tree generation step, it is plausible that better linearizations of the dependency tree might be achievable using more sophisticated techniques [8].

Chapter 3

Results And Conclusion

3.1 Evaluation

Evaluating sentence fusion is a notoriously difficult task [7] with no accepted competitive quantitative metrics. We have to depend on human judges for evaluation. Our system is compared 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 [7]; it is designed to measure the loss in grammaticality due to our linearization technique.

Native English speakers rated the fused sentences with respect to readability and content on a scale of 1 to 5 .A scoring rubric based on [17] is used. 12 judges participated in the study, for a total of 1062 evaluations[1]. The retained regions were boldfaced during the evaluation and the judges were instructed to base their content score on how well information was retained. For each set of inputs, each judge saw a single fusion drawn randomly from among the four systems. Results are displayed in Table 3.1.

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

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 3.1: Results of human evaluation

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

Table 3.2: Number of times each **Content** score was assigned by human judges.

3.2 Discussion

Readability scores clearly 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, while the remaining systems are all based on human-authored text, which is guaranteed to be grammatical. 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.

Human-authored sentences slightly outperform the readability upper bound on content. This shows that linearization has a negative effecto on content as both systems are trying to output the same text. Our system is slightly better than “and”-splicing. The distribution of scores is shown in Table 3.2. The system gets more perfect scores, but sometimes gets really low ones too. There is much less variance for “and”-splices.

Our system, while not achieving human performance, does not lag behind by a large amount in both metrics. It performs really well on some relatively hard sentences and gets most easy fusions right most of the time. For instance, the output on our example sentence shown in section 2.1 is exactly matching the oracle.

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

Input Sentences :

1. Clinton could erupt in red-faced rage.

2. George. W. Bush had his Texas swagger.

System Output:

Clinton could erupt in red-faced rage and George. W. Bush had his Texas swagger.

The "and"-splice baseline adds extraneous content in many cases. For the following example, the "and" baseline retains all of the second sentence. While our system also fuses the two sentences with "and", it correctly omits the second clause of the second sentence. Since there is no information selected for retention in that clause (as determined by the LCS alignment) and it is syntactically independent of the first clause, the system decides that it can be pruned. The journalist's fusion relies on recognizing a temporal discourse relationship between the two clauses.

Input Sentences :

1. Uribe appeared unstoppable in July after the army's rescue of French-Colombian politician Ingrid Betancourt and 14 other high-profile hostages held for years by guerrillas.

2. **His popularity shot to over 90 percent**, but since then news has been bad for the bespectacled cattle rancher considered Washington's closest South American ally.

Editor's fusion:

Uribe appeared unstoppable in July when his popularity shot to over 90 percent after the army rescued French-Colombian politician Ingrid Betancourt and 14 other high-profile hostages held for years by leftist guerrillas.

System Output:

Uribe appeared unstoppable in July after the army rescue of French-colombian politician Ingrid Betancourt and 14 other high-profile hostages held for years by guerrillas and his popularity shot to 90 percent.

Because of examples like this, the "and"-splice baseline is qualitatively dissimilar to human performance across the dataset. 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.

Our integration of node alignment into our solution procedure helps the system to find good correspondences between the inputs. In the following example, 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; the editor makes the same decision in their fused sentence.

Input Sentences :

1. The suit **claims** the company **helped fly terrorism suspects abroad to secret prisons**.

2. 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**.

Editor's fusion:

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.

System Output:

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.

In many cases, even when the result is awkward or ungrammatical due to poor linearization, the ILP system makes reasonable choices of mergers and dependencies to retain. In the following example, the system decides “Secretary-General” belongs as a modifier on “de Mello”, which is in fact the choice made by the editor. However, poor ordering creates the improper phrase “de Mello’s death who”.

This example also demonstrates that, when the LCS-aligned retained regions do not form a grammatical sentence, the system nonetheless keeps enough extra material to make the output intelligible. The words “could have been” are included in order to connect “Secretary-General” to “he”, despite the fact that the editor chose not to use them.

Input Sentences :

1. 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.

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

Editor’s fusion:

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.

System Output:

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

Similarly, the next example is a case of bad fusion due to faulty linearization– a correct linearization of the output tree would have begun “Vice President-elect Joe Biden, a veteran Democratic senator from Delaware who had contacted to lobby...” (the object of “contacted”, the word “her”, is omitted because of an unrelated syntactic analysis error).

System Output:
Biden a veteran Democratic senator from Delaware that Vice president-elect and Joe had contacted to lobby was quoted by the Huffington Post as saying Obama had made a mistake by not consulting Feinstein on the Panetta choice.

Some errors do occur during the ILP tree extraction process, rather than linearization. In this example, the system fails to mark the arguments of “took” and “position” as required, leading to their omission, which makes the output ungrammatical.

System Output:
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.

3.3 Conclusion

The supervised method presented here produces highly readable sentences when applied to naturally occuring data. It is clearly evident from the readability upper bound that the obvious improvement to make is in the linearizer. A more sophisticated linearizer might improve performance dramatically. However, as mentioned before, this work focuses more on content selection. Also, we try to avoid the problem of choosing the content to fuse as this is a very challenging discourse problem by itself.

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 [3].

While an automated pronoun coreference system might lead to a slight dip in performance, we do not expect it to be a great amount. Also, some bad fusions are due to parsing errors.

A straightforward task is to use this for fusion in single document summarization and finally, the system could also be trained for fusing similar sentences and we hope to make use of the new corpus of [16] for this purpose.

Bibliography

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

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

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

[4] Michael Collins. 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 2002.

[5] Hal Daume III and Daniel Marcu. 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 2004. Association for Computational Linguistics.

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

[7] Katja Filippova and Michael Strube. 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 2008. Association for Computational Linguistics.

[8] Katja Filippova and Michael Strube. 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 2009. Association for Computational Linguistics.

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

[10] Ilog, Inc. Cplex solver, 2003.

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

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

[13] Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 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 2008. Association for Computational Linguistics.

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

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

[16] Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 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 2010. Association for Computational Linguistics.

[17] Tadashi Nomoto. 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 2009. Association for Computational Linguistics.

[18] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 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 2004. Association for Computational Linguistics.

[19] Philip Resnik. 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, 1995. Morgan Kaufmann Publishers Inc.

[20] Jeffrey C. Reynar and Adwait Ratnaparkhi. 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., 1997.

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