Two-part Segmentation of Text Documents

Deepak P[1] Karthik Visweswariah[1] Nirmalie Wiratunga[2] Sadiq Sani[2]
[1]IBM Research - India, Bangalore, INDIA
[2]School of Computing, Robert Gordon University, Aberdeen, Scotland, UK
{deepak.s.p|v-karthik}@in.ibm.com {n.wiratunga|s.a.sani}@rgu.ac.uk

ABSTRACT

We consider the problem of segmenting text documents that have a two-part structure such as a problem part and a solution part. Documents of this genre include incident reports that typically involve description of events relating to a problem followed by those pertaining to the solution that was tried. Segmenting such documents into the component two parts would render them usable in knowledge reuse frameworks such as Case-Based Reasoning. This segmentation problem presents a hard case for traditional text segmentation due to the lexical inter-relatedness of the segments. We develop a two-part segmentation technique that can harness a corpus of similar documents to model the behavior of the two segments and their inter-relatedness using language models and translation models respectively. In particular, we use separate language models for the problem and solution segment types, whereas the inter-relatedness between segment types is modeled using an IBM Model 1 translation model. We model documents as being generated starting from the problem part that comprises of words sampled from the problem language model, followed by the solution part whose words are sampled either from the solution language model or from a translation model conditioned on the words already chosen in the problem part. We show, through an extensive set of experiments on real-world data, that our approach outperforms the state-of-the-art text segmentation algorithms in the accuracy of segmentation, and that such improved accuracy translates well to improved usability in Case-based Reasoning systems. We also analyze the robustness of our technique to varying amounts and types of noise and empirically illustrate that our technique is quite noise tolerant, and degrades gracefully with increasing amounts of noise.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text Analysis, Language Models*

Keywords

Text, Segmentation, Language Models, Translation Models

1. INTRODUCTION

Experience reports are prepared in various contexts for various purposes. An example is a bug report in software development which describes a sequence of events starting from events that characterize the occurence of the bug, and events involving fixing of the bug. Incident reports in other scenarios comprise descriptions of a temporal sequence of events, some of which characterize the cause of the incident, followed by those that describe the way the incident was dealt with. A real incident report from an airline company is shown below:

A Nexen employee checked his rebreather unit and found it to be past its "Service Due Date". He alerted the other passengers and they found the same situation with their re-breather units. All the units were promptly changed by the heliport staff and the flight proceeded as normal.

Of this, the first two sentences describe the (symptoms of the) incident, and the third indicates how the incident was resolved. In many cases, however, the events describing the *problem* and those describing the *solution* could be interleaved, especially at the boundaries. In clinical diagnosis, medical transcriptionists maintain diagnosis reports; these often have a similar two-part structure with a symptoms part and a diagnosis part.

School reports continuing difficulties with repetitive questioning. Obsession with cleanness on a daily basis. Inability to relate this well in the classroom. Asperger disorder. Obsessive compulsive disorder.

In the above sample medical transcription report (from MTSamples[1]), the first three sentences talk about symptoms whereas the remaining indicate diagnosis. In most cases where the diagnosis is complete, the two-part structure is very apparent. Root-cause Analysis (RCA) reports generated from service delivery organizations also have a two part structure (problem and solution).

Though such reports are mostly used currently for auditing and tracking purposes, it is easy to see that they provide a wealth of information that could be exploited in knowledge reuse systems such as Case-Based Reasoning[2]. CBR systems make use of problem-solution repositories to find possible solutions to new problems. For a newly posed problem, similar problems are retrieved from the repository and the solutions associated with them are deemed to be usable for the new problem. For example, a CBR system that uses symptom-diagnosis cases would respond to a new symptom report with historical diagnoses of similar symptoms and could help a medical practicioner arrive at a better/faster conclusion regarding the diagnosis. Depending on the complexity of the problems and solutions, linguistic processing would have to be performed to be able to choose/adapt the solutions of similar problems for us-

[1]http://www.mtsamples.com/
[2]http://en.wikipedia.org/wiki/Case-based_reasoning

age to solve the new problem. The CBR framework provides a powerful platform to exploit such experience and incident reports once they are partitioned into the problem and solution (or similar) components. Problem-solution partitioning is useful beyond CBR too; such paritioning enables better retrieval by usage of translation models [24].

In this paper, we address the problem of segmenting experience/incident reports into problem and solution parts. We will see that this is a hard case for traditional text segmentation approaches that typically rely on inter-segment *dissimilarity*; this is because the problem and solution parts may have significantly similar vocabulary due to being talking about related incidents. Towards solving our two-part segmentation problem, *we develop a technique that harnesses the availability of a set of similar incident reports to build language models (that learn the nature of problem and solution parts using statistical techniques) and translation models (that learn the correlation between words in the problem and those in the solution) that are then used to identify problem and solution segments*. We call our technique as Correlation and Cohesion Driven Segmentation (*CCS*). Our specific contributions are as follows:

- We propose a novel technique, *CCS*, that exploits a collection of similar two-part text documents to segment them.
- We present an empirical evaluation that depicts the superiority of our technique over state-of-the-art methods on segmentation quality by large and statistically significant margins.
- Further, we empirically illustrate that such improved segmentation by *CCS* leads to better solution qualities in a CBR system and that the *CCS* technique is robust to small amounts of noise in the input text.

Section 2 presents an overview of related work. We define our problem in Section 3 and present our approach in Section 4. Our experimental validation comprises Section 5 and we conclude in Section 6.

2. RELATED WORK

Segmenting experience/incident reports to problem and solution segments (or similar segmentation, e.g., cause-effect, symptoms-diagnosis) is a special case of the text segmentation problem [3]. Text segmentation involves splitting text into topical segments; the number or nature of the segments are usually not known a priori. Unsupervised text segmentation techniques may use domain-independent assumptions such as lexical cohesion [9] and/or domain dependent features such as cue words [7]. We briefly review text segmentation methods and knowledge reuse techniques that work on problem-solution data.

Text Segmentation, the problem of splitting text into *lexically coherent* or *topical* segments, started getting attention with the *TextTiling* algorithm [9]. Subsequent works focussed on using word repetetions [20], semantic networks [14], deep semantic features such as presence of co-reference across a candidate segment boundary [18] and hidden markov models [23] whereas segmental semi-markov models have been exploited for a related task of change-point detection [8]. A recent text segmentation algorithm, APS [12], uses affinity propagation to identify segment assignments. In contrast to the above approaches that deal with one document at a time, [15] present an approach that makes use of a collection of similar documents to segment individual documents better. *We will compare the approach that we develop against APS, TextTiling and [15].*

Using Problem-Solution Repositories: Case-based reasoning [13] or CBR relates to the theory and practice of reusing knowledge available as question-answer (or problem-solution) pairs; a problem-solution pair is referred to as a *case* in CBR parlance. CBR systems maintain a repository of cases, and respond to a new problem with solutions derived from those of problems related/similar to the new problem. Research in CBR focusses on improving the mechanisms to *retrieve* similar problems, *reuse* their solutions by adapting to solve the new problem, *revise* the adapted solution based on the new problem, and *retain* the new solution by storing it in the repository. In addition to the large recent interest in textual CBR [16, 2, 10], there has been a good amount of work in improving retrieval in textual question answer repositories [24, 11] of late, mostly from the information retrieval community. Usage of translation models to model question-answer vocabulary correlations was first explored in [6]. [24] builds a translation model using the question set and the answer set as a parallel corpus and uses it to conceptually 'expand' a new question before posing it to a retrieval system. Apart from being beneficial to knowledge reuse, such segmentation would help more traditional scenarios such as the task of a corporate knowledge-author who seeks to look through incident reports to identify and document new and interesting problems and their solutions.

3. PROBLEM DEFINITION

We now define the problem formally. Given a set $\mathcal{D}$ of n documents (e.g., incident/experience reports) of a similar nature, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$, with document $\mathcal{D}_i$ comprising of l_i words that we denote as $[w_{i\ 1}, \ldots, w_{i\ l_i}]$, we want to identify a vector, $\mathcal{Z}$, of n values $[z_1, z_2, \ldots, z_n]$ such that the segmentation of every document $\mathcal{D}_i$ into the two segments,

$$[w_{i\ 1}, \ldots, w_{i\ z_i}], [w_{i\ (z_i+1)}, \ldots, w_{i\ l_i}]$$

approximates as closely as possible the actual segmentation of the document $\mathcal{D}_i$ into its two inherent segments (e.g., problem and solution, symptoms and diagnosis etc.) that we will generically refer to as problem and solution segments from hereon. For every document $\mathcal{D}_i$, we will limit our search space to only those values of z_i that are sentence boundaries. In other words, we will ensure that sentences are not broken since it is not very natural to have parts of the sentence in the problem segment and the remaining in the solution. $dom(z_i)$ denotes all possible values of segmentation points (i.e., all segmentation points that are at a sentence boundary) for $\mathcal{D}_i$.

4. OUR APPROACH

In this section, we outline our approach for two-part segmentation of documents in a corpus of similar documents. We start by outlining certain assumptions that we use and go on to describe our approach in detail.

4.1 Assumptions

Our approach is based on some assumptions, some of which are well-known. At the risk of re-stating some of the obvious, we list the assumptions:

- **Separation:** We assume that problem and solution segments are reasonably well-separated. Such smoothness assumptions are at the core of most text segmentation approaches and is hence not novel. In particular, the worst case for our approach would be one where the problem and solution sentences are interleaved fully and the best case would be one where there is a contiguous sequence of problem sentences and another contiguous sequence of solution sentences, in the document.

- **Segment Ordering:** Unlike most other segmentation approaches, we assign a label to the segment (as either a problem or solution segment), and want to be able to reason across common segments across documents. To enable such reasoning, *we assume that the relative ordering of the two segments across documents in the corpus is reasonably consistent*. We argue that this assumption is not very restrictive since events are often described in chronological fashion in incident reports, and the events that are part of the problem typically precede those that form the solution. In other scenarios such as diagnosis reports, the symptoms are mostly described before the diagnosis.

- **Similar Problems have Similar Solutions:** The assumption that similar problems have similar solutions is the foundational principle for case-based reasoning [1]. At the word-level, this translates to the presence of correlations between certain words in the problem and certain others in the solution. We will use translation models to identify and use such correlations and use them in our segmentation algorithm.

- **Intra-segment Lexical Cohesion:** That sentences within the same segment have a higher likelihood of being lexically similar is a heavily used assumption in text segmentation literature (as seen in Section 2). Since we use a notion of a segment being either problem of solution, we extend the assumption to say that segments of the same type have a higher likelihood to be similar, *across documents*. We use language models to tap this assumption.

As in the case of any statistical/learning technique, we do not require each of the above assumptions to hold in an absolute sense; minor aberrations that do not throw the statistics haywire are typically easily tolerated. In Section 5.7, we will empirically evaluate the performance of our technique in datasets where some of these assumptions do not fully hold.

4.2 Correlation and Cohesion Driven Segmentation (CCS)

We now describe our approach, Correlation & Cohesion Driven Segmentation (CCS), that exploits the correlation between words in the problem and solution, to arrive at a segmentation of incident reports (or similar text documents) into problem and solution parts. Our approach is fully unsupervised and has no apriori knowledge of problem and solution behavior or the nature of correlatedness between them. CCS relies on the following premises that are based on assumptions in Section 4.1.

- A good segmentation of each of the documents can be used to learn problem and solution segment behavior well. Well-learnt solution and problem segment behavior can be used to derive a good segmentation.

- A good segmentation of each of the documents can be used to capture inter-segment word correlations better. Once the inter-segment word correlations are captured well, they could be used to derive a good segmentation.

We use the popular IBM Model 1 [4] translation model to learn word correlations across segments, and a unigram language model [22] to model behavior of segment types. We first describe the generative model for text documents that we use in our segmentation technique. We then outline an objective function and followed by the CCS approach which is basically an EM algorithm that optimizes the objective function.

4.2.1 The Generative Model

Consider a unigram problem language model $\mathcal{P}$, a unigram solution language model $\mathcal{S}$ and an IBM Model 1 translation model, $\mathcal{T}$, that models translation probabilities from problem words to solution words. The unigram language models for problems and solutions are multinomial distributions of words that would favor words that occur more in problems and solutions respectively. The translation model is intuitively a 2-d associative array with $\mathcal{T}[w][v]$ being directly related to the probability of the word v occuring in the solution whenever w is seen to occur in the problem; for consistency with previous literature on translation models, we will use $\mathcal{T}(v|w)$ to refer to this probability. Further, $\mathcal{T}(.|w)$ refers to the multinomial distribution of words with their value indicating the probability of occurence whenever w occurs in the problem.

Using such a set of models $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$, our generative model that generates a document $\mathcal{D}_i$ having the first z_i words in the problem and the remaining $(l_i - z_i)$ words in the solution, works as follows:

```
1. For $j, 1 \leq j \leq z_i$,
    (a) Choose $w_{i\ j} \sim Mult(\mathcal{P})$
2. For $j, z_i < j \leq l_i$,
    (a) Choose $r \sim U(0,1)$
    (b) If $r < \lambda$,
        i. Choose $w_{i\ j} \sim Mult(\mathcal{S})$
    (c) Else,
        i. Choose $w_{i\ j} \sim Avg(Mult(\mathcal{T}(.|w_{i\ 1})), \ldots, Mult(\mathcal{T}(.|w_{i\ z_i})))$
```

Informally, the z_i words (those belonging to the problem part) are sampled from the problem language model. The remaining $(l_i - z_i)$ words that are associated with the solution are then sampled from the solution langauge model with a probability of λ and from the average of the multinomial translation model distributions corresponding to each problem word with a probability of $(1 - \lambda)$.

Table 1: Example Report Document

Disk full reported.
Some files were deleted to resolve the issue.

Illustrative Example: Consider a hypothetical two sentence document from an IT helpdesk-like scenario, as given in Table 3. Let the segmentation point be the one that splits it into these two sentences, the first sentence being the problem and the second being the solution. Now, among the words {*files, deleted, resolve, issue*} in the solution, it is intuitively likely that *resolve* and *issue* have high probabilities under the solution language model, especially if $\mathcal{D}$ is a dataset of IT helpdesk reports. On the other hand, *files* and *deleted* are likely to be better supported by the translation model that is conditioned on very related words such as *disk* and *full* that already appear in the problem part. This intuition of the dual origin of solution words is factored into the generative model by allowing it to sample solution words from either the solution language model or the combination of the translation model distributions that are conditioned over the chosen problem words.

Probability Computation: Under the generative framework outlined above, the probability of generating a document $\mathcal{D}_x$ of which the first z_x words belong to the problem, given the models $\mathcal{P}, \mathcal{S}, \mathcal{T}$ is denoted as follows:

$$p(\mathcal{D}_x, z_x|\mathcal{P}, \mathcal{S}, \mathcal{T}) = \prod_{1 \le i \le z_x} \mathcal{P}(w_{x\ i}) \quad \times$$

$$\prod_{z_x < i \le l_x} (\lambda\, \mathcal{S}(w_{x\ i}) + (1-\lambda)\, avg\{\mathcal{T}(w_{x\ i}|w_{x\ 1}), .., \mathcal{T}(w_{x\ i}|w_{x\ z_x})\})$$

where $\mathcal{P}(w)$ denotes the probability associated with w in the multinomial model $\mathcal{P}$. For each of the problem words, the probability of the word according to the problem language model is used in the product formulation. The solution word's contribution to the product, similar to that in the generative model, is modeled as a weighted sum of its probability from the solution language model and the average of its probabilities from the multinomial translation model distributions for each of the z_x problem words.

For a document corpus $\mathcal{D}$ and a segmentation vector $\mathcal{Z}$ (Ref. Sec 3), the probability of the $[\mathcal{D}, \mathcal{Z}]$ combination given the models $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$ is then estimated as a product of the probabilities of the separate $(\mathcal{D}_i, z_i)$ combinations:

$$L(\mathcal{P}, \mathcal{S}, \mathcal{T}; \mathcal{D}, \mathcal{Z}) = p(\mathcal{D}, \mathcal{Z}|\mathcal{P}, \mathcal{S}, \mathcal{T}) = \prod_{\mathcal{D}_i \in \mathcal{D}} p(\mathcal{D}_i, z_i|\mathcal{P}, \mathcal{S}, \mathcal{T})$$

4.2.2 *The Objective Function and EM Overview*

For a set of documents $\mathcal{D}$, we can now estimate the models $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$ that are most likely to generate the document set. Towards this intent, our objective function that is to be maximized denotes the maximum likelihood estimate of the models.

$$L(\mathcal{P}, \mathcal{S}, \mathcal{T}; \mathcal{D}) = p(\mathcal{D}|\mathcal{P}, \mathcal{S}, \mathcal{T})$$

Since the segmentation vector $\mathcal{Z}$ is unknown, the maximal likelihood estimate is calculated by marginalizing over all possible values of $\mathcal{Z}$ (i.e., all possible segmentation points over all documents).

$$p(\mathcal{D}|\mathcal{P}, \mathcal{S}, \mathcal{T}) = \sum_{\mathcal{Z}' \in domain(\mathcal{Z})} p(\mathcal{D}, \mathcal{Z}'|\mathcal{P}, \mathcal{S}, \mathcal{T})$$

$$= \prod_{\mathcal{D}_i \in \mathcal{D}} \sum_{z \in dom(z_i)} p(\mathcal{D}_i, z|\mathcal{P}, \mathcal{S}, \mathcal{T})$$

We will use an iterative EM formulation [5] to find the maximum likelihood estimate outlined above. Each iteration is a sequence of two high-level steps, the E and M steps, that are summarized as follows:

- **E-step:** In the E-step, we use the current estimates of the language and translation models to compute the posterior probability of each segmentation point, for every document. At the document corpus level, this translates to determining the posteriors associated with each possible value of the segmentation vector $\mathcal{Z}$.
- **M-step:** The M-step re-builds the language and translation models in accordance with the posterior probabilities of various values of $\mathcal{Z}$.

In addition to the high-level steps outlined above, the E and M steps involve more detailed processing, especially, those corresponding to estimating whether solution words in each document were derived from the language or translation model and using them in re-estimating the models. We describe these steps in greater detail in respective sections.

4.2.3 *E-Step*

According to our generative model and the objective function derived from it, the maximum likelihood estimates of the models may be obtained if the following are known:

- The correct segmentation point for each document, $\mathcal{D}_i$
- Information as to the source of each of the solution word in the solution part of $\mathcal{D}_i$ i.e., whether it came from the solution language model or the translation language model

However, none of these information is available to us. Thus, in the E-step, we estimate this information using the current estimates of language and translation models. For each segmentation point in a document $\mathcal{D}_i$, we estimate the probability of that being the correct segmentation point. Similarly, for each such segmentation point, we estimate the probabilities that each solution word is derived from either of the sources. These estimates are then used in the M-step to rebuild the language and translation models, as we will show in the next section. We will use θ as a shorthand to denote the set $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$ for notational convenience, whenever appropriate. The estimated values for every $(\mathcal{D}_i, z')$ pair, where z' denotes any possible segmentation point for $\mathcal{D}_i$, are:

- $p(z'|\mathcal{D}_i, \theta)$: This denotes the posterior probability of the segmentation point z' according to the language and translation models for the document $\mathcal{D}_i$.
- Every word w in the solution part of $\mathcal{D}_i$ according to the segmentation point z' could have been derived from either the solution language model or the translation model. We determine these separate probabilities:
  - $p(Source = \mathcal{S}|w, \mathcal{D}_i, z', \theta)$ represents the probability that w in the solution part of $\mathcal{D}_i$ (according to the segmentation point z') was derived from the solution model.
  - $p(Source = \mathcal{T}|w, \mathcal{D}_i, z', \theta)$ analogously denotes the probability of the source being the translation model. Since the source needs to be either $\mathcal{S}$ or $\mathcal{T}$, these two values would always sum to 1.0.

Estimating Segmentation Point Posteriors: The posterior probability of each candidate segmentation point z' in a document $\mathcal{D}_i$ is obtained easily by conditioning the distribution $p(\mathcal{D}_i, z'|\theta)$ (computed as shown in Section 4.2.1) over the document $\mathcal{D}_i$:

$$p(z'|\mathcal{D}_i, \theta) = \frac{p(\mathcal{D}_i, z'|\theta)}{\sum_{z \in dom(z_i)} p(\mathcal{D}_i, z|\theta)}$$

Estimating Posteriors for Solution Word Source: The probability of the language model and translation model generating a given solution word w are assessed separately as follows:

$$p(\mathcal{S}, w|\mathcal{D}_i, z', \theta) = \lambda \times \mathcal{S}(w)$$

$$p(\mathcal{T}, w|\mathcal{D}_i, z', \theta) = (1-\lambda) \times avg\{\mathcal{T}(w|p)|p \in problem(\mathcal{D}_i, z')\}$$

The above construction is derived from the generative framework; λ is the relative weighting used in Section 4.2.1 whereas

$problem(\mathcal{D}_i, z')$ denotes the set of words in the problem when the document $\mathcal{D}_i$ is split at the segmentation point z'. The above two values when conditioned over w, give the posterior probabilities of the word w being generated from either sources.

$$p(Source{=}\mathcal{S}|w,\mathcal{D}_i,z',\theta){=}\frac{p(\mathcal{S},w|\mathcal{D}_i,z',\theta)}{p(\mathcal{S},w|\mathcal{D}_i,z',\theta)+p(\mathcal{T},w|\mathcal{D}_i,z',\theta)}$$

$$p(Source = \mathcal{T}|w,\mathcal{D}_i,z',\theta) = 1.0{-}p(Source = \mathcal{S}|w,\mathcal{D}_i,z',\theta)$$

4.2.4 M-Step

In the M-Step, we use the segmentation point posteriors and solution word souce assessments from the E-step to (re-)estimate the language and translation models.

Estimating the Language Models: We start with an overview on generating unigram language models. Let $\mathcal{V}_1 = [\{w_1 = 0.8, w_2 = 0.3\}, 0.4]$ and $\mathcal{V}_2 = [\{w_1 = 1.2\}, 0.5]$ denote two conceptual documents; the first one contains the word w_1 with a frequency of 0.8 and w_2 with a frequency of 0.3 whereas the second one contains just one word w_1 with a frequency of 1.2. The second entry in each document representation is meant to represent the weight associated with the document; in this case, the weights for the documents are seen to be 0.4 and 0.5 respectively. Though documents do not have fractional word frequencies in reality, our statistical estimates can accomodate such fractional frequencies and weights. The unigram language model derived out of a collection of such documents is a multinomial distribution where the value corresponding to any word w is computed as follows:

$$\mathcal{L}(w) = \frac{\sum_{\mathcal{V}_i} weight(\mathcal{V}_i)\ freq(\mathcal{V}_i, w)}{\sum_{w'}\sum_{\mathcal{V}_i} weight(\mathcal{V}_i)\ freq(\mathcal{V}_i, w')}$$

where w' is any word in the vocabulary and the $weight(.)$ and $freq(.,.)$ functions denote the document weight and document-specific word frequency respectively. In a language model generated from $\mathcal{V}_1$ and $\mathcal{V}_2$, the value corresponding to w_1 would be:

$$\mathcal{L}(w) = \frac{0.4*0.8+0.5*1.2}{(0.4*0.8+0.5*1.2)+(0.4*0.3)} = 0.8846$$

For each segmentation point z' for every $\mathcal{D}_i$, let $Prob(\mathcal{D}_i, z')$ and $Sol(\mathcal{D}_i, z')$ denote the set of words in the problem and solution parts respectively. We generate one conceptual document each for the problem and solution part (denoted as $\mathcal{V}_\mathcal{P}(.)$ and $\mathcal{V}_\mathcal{S}(.)$ respectively) as follows:

$$\mathcal{V}_\mathcal{P}(\mathcal{D}_i, z') = [\{w = 1.0|w \in Prob(\mathcal{D}_i, z')\}, p(z'|\mathcal{D}_i, \theta)]$$

$$\mathcal{V}_\mathcal{S}(\mathcal{D}_i, z') =$$

$$[\{w{=}p(Source{=}\ \mathcal{S}|w,\mathcal{D}_i,z',\theta)|w \in Sol(\mathcal{D}_i, z')\}, p(z'|\mathcal{D}_i,\theta)]$$

Informally, the problem document is simply the collection of problem words in $Prob(\mathcal{D}_i, z')$ with the document weight being the posterior probability of the segmentation point. The solution document is also weighted by the posterior probability of the segmentation point; however, unlike the problem case, each word in the solution document has a frequency that is determined by the probability of it being generated by the solution model. Each document $\mathcal{D}_i$ thus generates as many problem and solution documents as there are possible segmentation points (i.e., $|dom(z_i)|$) in it. The problem documents collection is used to estimate $\mathcal{P}$ whereas $\mathcal{S}$ is generated from the collection of solution documents; these are models in conformance with the estimates derived in the E-step.

Estimating the translation model: The translation model represents the correlation between words in the problems and those in the solutions. Towards this, it makes use of a corpus of document pairs such as below:

$$[\{w_1 = 0.8, w_2 = 0.4\}, \{w_1 = 0.2, w_3 = 0.6\}, 0.8]$$

The example above denotes a document pair as a 3-tuple, with the first element denoting the frequencies of words in the problem part, second denoting the frequencies of words in the solution part whereas the third element denotes the weight assigned to the document pair. Towards estimating the translation model, we create one such 3-tuple for each $(\mathcal{D}_i, z')$ pair as follows:

$$[\{w = 1.0|w \in Prob(\mathcal{D}_i, z')\},$$

$$\{w{=}p(Source = \mathcal{T}|w,\mathcal{D}_i,z',\theta)|w \in Sol(\mathcal{D}_i, z')\}, p(z'|\mathcal{D}_i,\theta)]$$

Analogous to the language model case, each document would generate as many 3-tuples as there are segmentation points. The collection of such 3-tuples are then used to learn a translation model which would then be in accordance with the E-step estimates. In particular, we use the weighted document pairs to derive an IBM Model 1 using a straightforward adaptation of the EM algorithm from [4]; we do not delve into the finer details of the translation model training process since that is tangential to the focus of this paper.

Alg. 1 ***CCS***

```
Input. $\mathcal{D}$, a set of documents
Output. $\mathcal{Z}$, a vector denoting the segmentation

1. Segment documents in $\mathcal{D}$ using
     state-of-the-art techniques to initialize $\mathcal{Z}$
2. Estimate the models $\mathcal{P}$, $\mathcal{S}$ and $\mathcal{T}$ using $\mathcal{Z}$
3. while ($p(\mathcal{D}|\mathcal{P},\mathcal{S},\mathcal{T})$ has not yet converged)
4.     E-Step: Estimate the segmentation point posterior
           probabilities and solution word source probabilities
5.     M-Step: Re-estimate the language and translation
           models using the E-step probabilities
6. $\forall \mathcal{D}_i \in \mathcal{D}$
           $z_i = \underset{z \in dom(z_i)}{\text{argmax}}\ p(\mathcal{D}_i, z|\mathcal{P},\mathcal{S},\mathcal{T})$
7. return $\mathcal{Z}$
```

4.2.5 The CCS Algorithm

The CCS algorithm is outlined in Algorithm 1. We start in Line 1 by initializing the $\mathcal{Z}$ vector according to the segmentation derived from a state-of-the-art segmentation algorithm (e.g., APS [12], [15] or TextTiling [9]); we will use APS to initialize the segmentation in our experiments (Ref. Section 5.4). Segmentations by generic text segmentation algorithms, however, do not necessarily generate exactly two segments per document; we use a post-processing step (in Line 2) to convert any multi-segmentation to a two-part segmentation by retaining only the most appropriate segment switch for each document, as determined using a TextTiling-style estimation[3]. The models are built using the initialized $\mathcal{Z}$ and an iterative sequence of

[3]TextTiling estimates the score of each sentence boundary to be a

Table 2: Datasets and Sizes

Dataset	#Docs	#Sents per doc	#Words per doc
visa	511	4.22	49.75
health	433	5.98	72.61
agri	229	5.10	70.38
loan	514	4.90	61.88
tourism	134	4.79	60.95
railways	268	4.51	50.57
telecom	103	4.70	47.48
web	109	4.08	49.27

E-step (Ref. Section 4.2.3) and M-step (Ref. Section 4.2.4) operations follow. We run this sequence of steps until no more changes occur to the objective function in Section 4.2.2, or for 20 iterations, whichever is fewer.

When the iterations are complete, we set the segmentation point for each document as that which maximizes the probability of generating that document, according to the final estimates of the language and translation models. These form the $\mathcal{Z}$ vector that is then output as the final segmentation for the documents. Since there is no apriori evidence to guess the relative importance of the solution language model and the translation model in generating solution words, we weigh them equally by setting $\lambda = 0.5$ in our approach. In the remainder of this paper, unless mentioned otherwise, *CCS* refers to this setting of λ.

Time Complexity: Consider a corpus of *n* documents with a vocabulary of size *m*, each document having an average of *l* sentences or *w* words. Calculating the posterior probabilities in the E-step costs $\mathcal{O}(lnw^2)$. The M-step operations of learning the translation models (using *k* iterations) and language models costs $\mathcal{O}(k(nw^2 + m))$ and $\mathcal{O}(nw + m)$ respectively. For k' iterations of *CCS*, the total time taken, hence, is of the order of $\mathcal{O}(k'nw^2(k + l) + k'km)$.

5. EXPERIMENTAL EVALUATION

We now describe our experimental study where we compare the CCS algorithm against the state of the art algorithms. We describe the datasets, evaluation measures and baseline algorithms followed by a detailed description of our extensive experimentation.

5.1 Datasets

In the absence of any available segmented incident report data (to the best of our knowledge), we use various datasets that were collected as part of a recent IR task and are publicly available[4]. We selected 8 domains from the training dataset preferring those domains that comprise verbose descriptions of problems and associated solutions. These datasets, unlike typical experience/incident reports, often have a first person narrative in the problem part (e.g., *I have been in need of a career switch to something related to networking and have a UK work visa*), and a slightly instructional narrative at the solution part (e.g., *Most UK companies look for a work visa that is valid for beyond one year*). It may be noted that such style differences between segments are advantageous for all text segmentation techniques (including baselines); since *CCS* is not tapping such style differences explicitly, we argue that it is kosher to attribute its improved performance with respect to the baselines mostly to the *CCS* formulation. The style differences may be captured in the intra-segment lexical cohesion assumption to some extent; while lexical cohesion of segments at the document level is used by all the baselines that we compare against, *CBA* (Ref. Sec 5.3) exploits the lexical cohesion of the same segment type across documents in the corpus. The various datasets (named by their domains) are listed in Table 2 and example problem-solution pair is given in Table 3. The documents in our datasets are seen to have 4-6 sentences and 45 to 70 words, on an average. We collate the problem and solution part to create a single document, and run the segmentation algorithms on them; the quality of segmentation is then evaluated with respect to the actual segment boundary (which is known, since the collation of parts to arrive at the single document was performed by us).

5.2 Evaluation Measures

We now outline the various evaluation measures that we use in our empirical study:

WindowDiff: We primarily use the well-known segmentation evaluation measure, *WindowDiff* [19], to evaluate segmentation quality. *WindowDiff* can be conceptually thought of as moving a sliding window simultaneously over the two segmentations (the created, and the ground truth), capturing the differences in the number of segment boundaries at each step, and then aggregating it across the entire document to arrive at a single measure of segmentation agreement. Consider two segmentations $\mathcal{Z}_1$ and $\mathcal{Z}_2$ of a document comprising of l sentences; the *WindowDiff* metric is computed as follows:

$$WD(\mathcal{Z}_1, \mathcal{Z}_2) = \frac{\sum\limits_{1 \leq i \leq (l-w+1)} f(\#SB(\mathcal{Z}_1, i, w), \#SB(\mathcal{Z}_2, i, w))}{l - w + 1}$$

where $\#SB(\mathcal{Z}, i, w)$ counts the number of segment boundaries according to the segmentation $\mathcal{Z}$ among the w sentences from the i^{th} sentence in the document, and $f(.,.)$ is a function that returns 1 if the two arguments are equal, and 0 otherwise. w is typically chosen as half of the average segment size. The *WindowDiff* values are then averaged across documents in the corpus. It may be noted that *WindowDiff is a penalty measure with lower values indicating better agreement among the segmentations compared.*

P_K**:** This metric [3], the precursor to *WindowDiff*, is very similar to the latter in using a sliding window type approach. Instead of counting the number of segment boundaries within a sliding window for each segmentation, P_K checks whether the two ends of the window are in the same segment. For a window, in cases where the segmentations disagree in terms of the membership of the sentences in the two ends, a penalty of 1.0 is added to the numerator. These are then averaged across sliding windows and documents similar to that in the *WindowDiff* measure.

Diff: *WindowDiff* and P_K are metrics that can handle segmentations that segment documents into any number of segments. However, our problem deals with choosing just one segmentation boundary for each document, where it would be segmented into two parts. *Diff* is a simple measure that measures the distance between the boundaries chosen in the segmentations that are compared. For example, for a document in question, if $\mathcal{Z}_1$ chooses a segmentation point beyond the z_1^{th} sentence and z_2 denotes the choice by $\mathcal{Z}_2$, the $Diff(\mathcal{Z}_1, \mathcal{Z}_2)$ is estimated as $abs(z_1 - z_2)$, with $abs(.)$ denoting the absolute value. This is a very intuitive measure since it gives the number of sentences that a segmentation is *off* by, when compared with the ground truth.

segmentation point and uses a threshold to designate all points that have a higher assessed score as segmentation points. In our adaptation to derive a two-part segmentation from a multi-segmentation, we simply choose that candidate among the multiple segmentation points that scores best (according to the TextTiling estimate), as the only segmentation point.

[4]http://www.isical.ac.in/ clia/faq-retrieval/faq-retrieval.html

Table 3: Example Problem & Solution from *loan*

My home was appraised by VA and now I am having problems with its condition. Since the appraisal is an inspection of the property, I think the VA should be able to help me with the problems.
Although the VA fee appraiser must view the property from both the exterior and interior to determine its overall condition, the appraisal process is not intended to be an "inspection" of the property. . . .

CBR Usability Measures: Since our eventual goal is to aid knowledge reuse, we also illustrate the improvements achieved on a CBR system that uses the CCS segmented dataset with respect to the one that uses other segmentations; in particular, we use the max and tot [17] metrics; more details are in Section 5.5.

Statistical Significance: We also present results of statistical significance on various measures using randomization tests [21] with a p-value of < 0.05. A technique being statistically significant over another with a p-value of < 0.05 suggests that the probability that the former achieved superior results by mere chance is less than 0.05; such statistical significance tests are becoming standard practice in evaluating retrieval systems[5].

5.3 Baseline Approaches

TextTiling [9] (TT): Among the earliest algorithms for text segmentation, TextTiling relies on lexical frequency and distributional information to identify segment boundaries. This uses the cosine similarity between two blocks of text, one before and another after each candidate segment boundary, to determine whether a segment boundary be placed at the location. We adapt *TextTiling* by forcing it to choose only one segment boundary per document. This is trivial since *TextTiling* scores each sentence boundary; instead of using a threshold to choose possibly multiple candidate sentence boundaries as segmentation points, we simply choose the single sentence boundary with the best score as the segmentation point.

Clustering-based Approach [15] (CBA): This, unlike most other algorithms in literature, is similar to *CCS* in the usage of knowledge across a corpus of similar text documents to segment each document. It clusters sentences across documents to arrive at sentence clusters, which are then clustered using spatial similarity (in the documents that contain them) to arrive at larger clusters called *representative segments*. Each document is then segmented using such *representative segments*. We adapt this approach by running the spatial-similarity based clustering until only two *representative segments* remain. Under such a setting, given a corpus with two segment types that manifest in many documents in the corpus, the clustering process in *CBA* is expected to produce one representative segment per segment type. This representative segment model, due to being built across documents, enables *CBA* to use corpus level signatures per segment type to segment each document; such a corpus-level modeling of segment types is something that algorithms that deal with one document at a time are incapable of doing. Even with two representative segments, CBA could produce multipe segment boundaries; in such case, we use *TextTiling* type scoring of such segment boundaries to choose the best segment boundary as the segmentation point.

APS [12]: In this recent algorithm, each basic unit (e.g., sentence) is treated as a data point. Similarities between data points are estimated using lexical measures unless provided by other means.

[5]http://faculty.vassar.edu/lowry/ch4pt1.html

Table 4: WindowDiff Evaluation

Dataset	TT	CBA	APS	CCS$_\mathcal{L}$	CCS
visa	0.307	0.483	<u>0.216</u>	0.158	**0.085**$\star$
health	0.379	0.309	<u>0.302</u>	0.106	**0.030**$\star$
agri	0.329	0.419	<u>0.200</u>	0.135	**0.052**$\star$
loan	0.369	0.402	<u>0.261</u>	0.223	**0.042**$\star$
tourism	0.373	0.398	<u>0.215</u>	0.188	**0.079**$\star$
railways	0.341	0.380	<u>0.213</u>	0.138	**0.015**$\star$
telecom	0.374	0.380	<u>0.276</u>	0.269	**0.186**$\star$
web	0.366	0.480	<u>0.213</u>	0.147	**0.018**$\star$
Average	0.355	0.406	<u>0.237</u>	0.170	**0.063**

An iterative message passing based on the affinity propagation formulation is then run, until coherent segments (sets of basic units) emerge. This approach could generate segmentations that have more than two segments per document. Similar to earlier scenarios, for such cases, we choose the best segment break from among them using a *TextTiling* style evaluation.

5.4 Segmentation Quality Evaluation

5.4.1 WindowDiff Evaluation

For each technique, we assess the segmentation generated using the *WindowDiff* measure when compared against the ground truth segmentation, and present the results in Table 4. *WindowDiff* being a penalty measure, the technique scoring lesser is considered as being better. Among the two types of models that *CCS* uses, the language and translation models, the former is more intuitive since it stems from the intra-segment lexical cohesion assumption that is central to most text segmentation algorithms. The language model representation of each segment type is analogous to the representative segment representation in the *CBA* approach. Towards illustrating the value of the usage of the translation model over and above the language models, we include the results of the *CCS* variant that uses only language models. This corresponds to setting $\lambda = 1.0$ in the *CCS* approach; we refer to this variant as $CCS_\mathcal{L}$ and include the results derived from it in Table 4.

The results table presented in Table 4 illustrates the effectiveness of the *CCS* formulation. The best among the baseline approaches is indicated by an underline. *APS* is seen to outperform the other baselines significantly, and was thus chosen to initialize the $\mathcal{Z}$ vector in Line 1 of Algorithm 1. *CCS* beats the baselines by large margins (the best number for each dataset is indicated in boldface) and also fares much better than *CCS*$_\mathcal{L}$ on each dataset; this illustrates that the translation models help improve the segmentation considerably over and above the usage of language models alone.

Statistical Significance ($p < 0.05$): Those entries of *CCS* that were found to be statistically significant over the corresponding entries of each of the other techniques (including CCS_L) are marked with a $\star$. A seen from the Table, *CCS* performance is statistically significant over every technique on each dataset.

5.4.2 P_K and Diff Evaluation

We now analyze the performance of the techniques on the P_K and *Diff* penalty measures. Since the general trends were similar to the *WindowDiff* evaluation with *APS* outperforming the other baselines, we present a comparison between *CCS* and *APS* herein.

As seen from Figure 1, *CCS* is able to bring down the P_K values down by upto three times than that of *APS*. When averaged across datasets, *APS* and *CCS* were seen to score 0.27 and 0.09 respec-

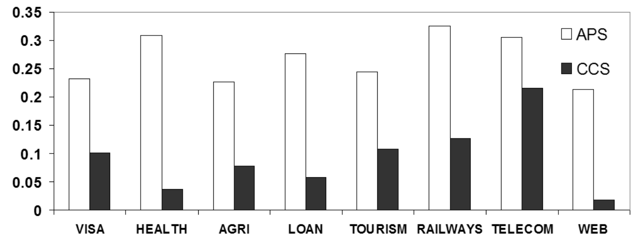

Figure 1: *APS* **and** *CCS* **on the** $\mathbf{P}_K$ **Measure**

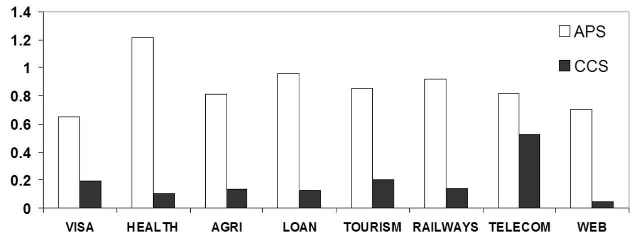

Figure 2: *APS* **and** *CCS* **on the** *Diff* **Measure**

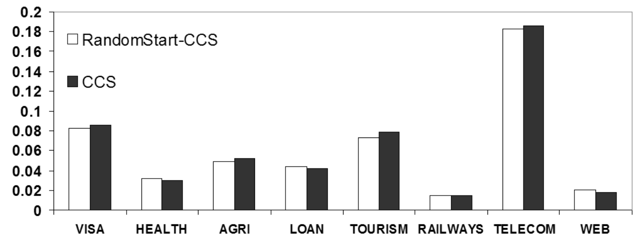

Figure 3: *CCS* **initialization Analysis**

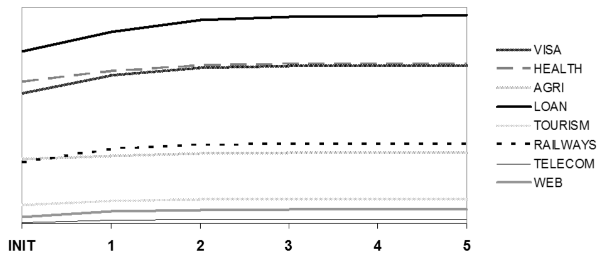

Figure 4: *CCS* **Objective Function Across Iterations**

tively. These results were seen to be statistically significant on each dataset at a p-value of less than 0.05. The efficacy of *CCS* over *APS* was seen to be much more pronounced under the *Diff* measure with the techniques scoring 0.19 and 0.87 respectively. The chart appears in Figure 2. Informally, the *CCS* segmentation was off from the ground truth by 0.19 sentences on an average, whereas the *APS* segmentation points were seen to be as much as 0.87 sentences away from the ground truth segmentation. Under the *Diff* measure too, the *CCS* performance was seen to be statistically significant over *APS*.

5.5 CBR Usability Evaluation

Though more accurate segmentation would intuitively be expected to deliver a better performance when used in a CBR system, we use a more direct measure to quantify the actual improvement. The tot measure [17] uses a leave-one-out style evaluation by posing each problem against the repository and measures the similarity between the top-k retrieved solutions against its own (which are known) using cosine similarity. Much like the configuration used in [17], we use cosine similarity as a proxy for the usability of the solution and set k=3. tot then denotes the denotes the total usability of the top-k solutions that are retrieved. The tot measure is to be understood as a lower bound on the usability, since the cosine similarity cannot model polysemy, and linguistic processing that could help rendering the retrieved solution more appropriate. The tot (for which, a higher value indicates better performance, unlike *WindowDiff*) value across the datasets for the *APS* and *CCS* techniques is presented in Table 5. As usual, the best measure is shown in boldface and *CCS* segmentation is seen to yield a better CBR system consistently, providing an average of **7%** gains across datasets. Similar results were obtained on the *max* measure [17] also. Much like in Table 4, we indicate statistically significant results by a $\star$; from Table 5, it is observed that the usability improvements achieved by *CCS* are statistically significant over *APS* on 5 datasets at a p-value < 0.05.

Table 5: CBR Usability Evaluation (*tot* measure)

Dataset	APS	CCS
visa	0.238	**0.252**$\star$
health	0.268	**0.296**$\star$
agri	0.394	**0.395**
loan	0.369	**0.404**$\star$
tourism	0.161	**0.162**
railways	0.546	**0.612**$\star$
telecom	0.202	**0.227**$\star$
web	0.236	**0.246**
Average	0.302	**0.324**

5.6 CCS Specific Analysis

5.6.1 CCS Initialization Analysis

Since we use the best-performing baseline, *APS*, to initialize the segmentation for the *CCS* technique (in line 1 in Algorithm 1), part of the credit for the good performance of the latter is likely to be due to the performance of *APS*. In this section, we analyze the robustness of the *CCS* formulation by subjecting it to a scenario where a good initialization is not available. In particular, we provide a random initialization of $\mathcal{Z}$ for *CCS* to start processing. Somewhat surprisingly, the randomly initialized *CCS* variant was seen to be highly competetive with that of the *APS* initialization. The *WindowDiff* measures are plotted in Figure 3; it is seen that the performance is nearly identical with minor variations. This shows that the *CCS* technique is extremely robust and can easily recover from bad initializations. However, a good initialization is likely to be more critical while working with datasets of longer documents that may span dozens of sentences each.

5.6.2 CCS Convergence

Due to the EM formulation, the value of the objective function outlined in Section 4.2.2 is bound to monotonically improve with each iteration. We now analyze the trends of the objective function values across EM iterations; Figure 4 plots the objective function

Table 6: Translation Model Samples from *railways*

Problem Word	*cancel*	*duplicate*	*delivery*
Correlated Solution Words	*tdr*	*deducted*	*letter*
	etktcanc	*misplaced*	*authority*
	slip	*authentic*	*authorization*
	printed	*genuineness*	*cities*

values in the Y-axis against the number of iterations in the X-axis. As expected, large gains are achieved between the initialization and the first iteration with the gains becoming smaller in subsequent steps. More importantly, the objective function is seen to stabilize very fast with the gains becoming marginal beyond the third iteration, on every dataset. Thus, *CCS* may be terminated beyond the third iteration in case of a need to optimize on computational expense, since the segmentations are likely to have been relatively stabilized.

5.6.3 Example Translation Model Estimates

Having shown that the usage of translation models in the *CCS* formulation leads to substantially more accurate segmentations, we now present a few sample translation model correlations to illustrate the kind of correlations that are learnt by it. We focus on the *railways* dataset, documents within which mostly start with descriptions of specific scenarios in relation to ticketing and traveling in the Indian Railway network, followed by steps to overcome such situations[6]. We pick three words that are commonly found in problem parts in the *railways* dataset based on a cursory glance through the dataset; these are *cancel*, *duplicate* and *delivery*. For each chosen problem word, we mine for the top-10 solution words that are correlated with it, and present a sample from them in Table 6.

We briefly outline intuitions as to why the correlations in Table 6 may be meaningful. **cancel** was found to be associated with situations involving ticket cancellation, for which remedies included resorting to one of the *tdr* or *etktcanc* cancellation processes provided by the railway. The third option, that of walking in to the counter for cancellation involves procuring a cancellation *slip* or *printing* it from the internet. Another common circumstance of interest is related to fetching a **duplicate** ticket. Towards getting this done, one needs to report that the ticket has been *misplaced* and produce ID proofs to assert the *authenticity* or *genuineness* of the request. Issuance of a duplicate ticket often involves *deduction* of a prescribed fee too. When tickets are booked online, Indian Railways provides an option of mailing a printed ticket to the passenger through courier. **delivery** of such tickets often is often problematic with common problems being unavailability of the passenger at the address location or due to the railways not providing delivery service in certain *cities*. The former problem is often easily resolved by leaving an *authorization letter* with someone who is available at the address provided for delivery.

5.7 Evaluation of Robustness to Noise

Unlike *APS* and other approaches that segment each document independently, *CCS* uses models created out of the entire corpus to segment each document. This, as illustrated above, leads to significant improvements in segmentation accuracy. However, it also implies that noise in a few documents in the corpus could affect the segmentation of even non-noisy documents in the corpus. In this section, we evaluate the robustness of *CCS* to two different kinds of noise to which incident reports are susceptible.

[6]The choice of dataset was also partially motivated by the domain knowledge of one of the authors; such considerations are inevitable is because understanding and making sense of word pairs is a fairly knowledge intensive process.

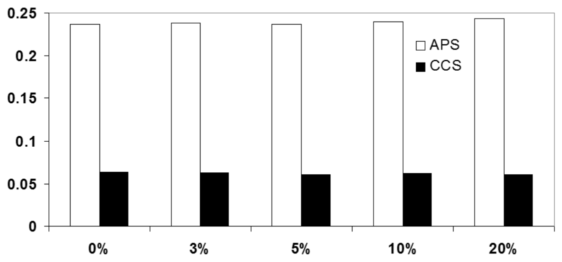

Figure 5: *WindowDiff* **under varying levels of** *Sentence Swapping*

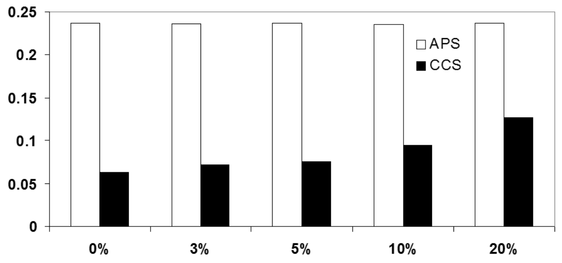

Figure 6: *WindowDiff* **under varying levels of** *Segment Swapping*

5.7.1 Interleaving at Boundaries

Incident reports often follow a chronological order of narration; however, an initial solution step could be followed by an incident that discovers something more about the problem, leading to a violation of the assumption of clear separation between the segments. To evaluate the robustness of *CCS* to this kind of noise, we inject such perturbation in a few documents by swapping the positions of sentences on either side of the true segment boundary (i.e., by making the last sentence in the problem segment as the first statement in the solution segment and vice versa); we call such noise as *Sentence Swapping*. We let *CCS* operate on the corpus that contains varying fractions of such noisy documents, and evaluate the performance with respect to *APS* on the *WindowDiff* measure. In accuracy evaluation using the *WindowDiff* measure, we consider only the *non-noisy* documents since the real segmentation for the noisy documents is not well-defined (due to the swapping).

We illustrate the *WindowDiff* evaluation in Figure 5. *APS* is not affected by the presence of noisy documents since it operates on a per-document level and the noisy documents are not considered in the *WindowDiff* computation as indicated above. More importantly, it is interesting to note that *CCS* performance is also seen to be very stable across varying fractions of noisy documents in the corpus; we show values for upto 20% noise (in the X-axis) in the chart.

5.7.2 Segment Ordering

Certain authors could violate the convention of segment ordering (e.g., problem segment followed by solution) and may use the opposite ordering. The presence of a few documents with the opposite ordering in the collection is detrimental to techniques like *CCS* that operate at the corpus level. Towards evaluating effects of such noise, we swap the ordering of problem and solution segments in a few documents (i.e., *Segment Swapping*). As in the earlier case, we evaluate the performance on corpora containing varying fractions of such noisy documents. Unlike *Sentence Swapping*, the boundaries are well defined even in the case of noisy documents since whole segments are swapped.

While *APS* is unaffected by such noise due to the per-document formulation, *CCS* is empirically seen to be fairly sensitive to swapping of whole segments in documents in the collection. It may be seen from the *WindowDiff* evaluation in Figure 6 that *CCS* degrades from 0.06 to 0.13 when the fraction of noisy documents is increased to up to 20%. However, even at 20% noise, *CCS* is seen to perform twice as better as *APS*, and is hence, still remains the preferred technique.

6. CONCLUSIONS AND FUTURE WORK

We outlined the two-part text segmentation problem to segment documents such as incident reports that contain problem and solution segments. Accurate identification of problem solution segments is critical towards making the knowledge in incident reports and diagnosis reports available to knowledge reuse systems. This problem, however, poses an unfriendly scenario to traditional text segmentation algorithms due to the relatedness of the segments. We outlined an iterative technique, *CCS*, that models the behavior of problem and solution segments and word-correlations across them in a corpus of similar documents, and exploits such modeling to arrive at more accurate segmentations. The *CCS* generative model formulates each document as being generated by initially sampling words from a problem language model, followed by choosing words from either the solution language model or a translation model conditioned on the words already chosen for the problem part. Our empirical study over a large collection of datasets establish that *CCS* provides vast and statistically significant improvements (on the *WindowDiff* measure) over state-of-the-art techniques, including techniques that use corpus-wide knowledge. Such improvements in segmentation accuracy are seen to reflect as improved solution usability in Case-based Reasoning systems. We have further shown that our technique can recover from not-so-good initializations, is reasonably noise-tolerant, and degrades very gracefully with increasing noise in the dataset. In short, we have established the utility of corpus-wide statistics and inter-segment word correlation in two-part text document segmentation, whenever a corpus of similar documents are available.

Since higher order language models (e.g., 2-gram, 3-gram etc.) and translation models (IBM Models 2 and beyond) are designed to rectify some of the drawbacks with the first order models, harnessing them to improve text segmentation would be an interesting future work. In this work, we have exploited the correlatedness of two segments; generalizing this into a framework that could detect and exploit word correlations in scenarios where any number of segments may be expected in a document, would be a useful extension to this work and a logical next step.

7. REFERENCES

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

[2] I. Adeyanju, N. Wiratunga, R. Lothian, and S. Craw. Applying machine translation evaluation techniques to textual cbr. In *ICCBR*, pages 21–35, 2010.

[3] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Mach. Learn.*, 34:177–210, February 1999.

[4] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16:79–85, June 1990.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[6] A. Echihabi and D. Marcu. A noisy-channel approach to question answering. In *ACL*, pages 16–23, 2003.

[7] J. Eisenstein and R. Barzilay. Bayesian unsupervised topic segmentation. In *EMNLP*, pages 334–343, 2008.

[8] X. Ge and P. Smyth. Segmental semi-markov models for change-point detection with applications to semiconductor manufacturing, 2000.

[9] M. A. Hearst. Multi-paragraph segmentation of expository text. In *ACL*, pages 9–16, 1994.

[10] K. Jayanthi, S. Chakraborti, and S. Massie. Introspective knowledge revision in textual case-based reasoning. In *ICCBR*, pages 171–185, 2010.

[11] J. Jeon, W. B. Croft, and J. H. Lee. Finding semantically similar questions based on their answers. In *SIGIR*, 2005.

[12] A. Kazantseva and S. Szpakowicz. Linear text segmentation using affinity propagation. In *EMNLP*, pages 284–293, 2011.

[13] J. L. Kolodner. An introduction to case-based reasoning. *Artif. Intell. Rev.*, 6(1):3–34, 1992.

[14] H. Kozima. Text segmentation based on similarity between words. In *ACL*, pages 286–288, 1993.

[15] K. Kummamuru, D. P, S. Roy, and L. V. Subramaniam. Unsupervised segmentation of conversational transcripts. In *SDM*, pages 834–845, 2008.

[16] M. Lenz, A. Hübner, and M. Kunze. Textual cbr. In *Case-Based Reasoning Technology, From Foundations to Applications*, pages 115–138, 1998.

[17] D. P, S. Chakraborti, and D. Khemani. More or better: on trade-offs in compacting textual problem solution repositories. In *CIKM*, pages 2321–2324, 2011.

[18] R. J. Passonneau and D. J. Litman. Discourse segmentation by human and automated means. *Computational Linguistics*, 23(1):103–139, 1997.

[19] L. Pevzner and M. A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[20] J. C. Reynar. An automatic method of finding topic boundaries. In *ACL*, pages 331–333, 1994.

[21] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. CIKM, pages 623–632, New York, NY, USA, 2007. ACM.

[22] F. Song and W. B. Croft. A general language model for information retrieval. CIKM, pages 316–321, New York, NY, USA, 1999. ACM.

[23] P. van Mulbregt, I. Carp, L. Gillick, S. Lowe, and J. Yamron. Text segmentation and topic tracking on broadcast news via a hidden markov model approach. In *ICSLP*, 1998.

[24] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482, 2008.