Automatic Generation of Question Answer Pairs From Noisy Case Logs

Jitendra Ajmera [#1], Sachindra Joshi [#2], Ashish verma [#3], Amol Mittal [4]

[#] *IBM India Research Lab*
New Delhi, India
[1] jajmera1@in.ibm.com
[2] jsachind@in.ibm.com
[3] vashish@in.ibm.com

[*] *Indian Institute of Technology*
New Delhi, India
[4] mt5100584@iitd.ac.in

***Abstract*—In a customer support scenario, a lot of valuable information is recorded in the form of 'case logs'. Case logs are primarily written for future references or manual inspections and therefore are written in a hasty manner and are very noisy.**

In this paper, we propose techniques that exploit these case logs to mine real customer concerns or problems and then map them to well written knowledge articles for that enterprise. This mapping results into generation of question-answer (QA) pairs. These QA pairs can be used for a variety of applications such as dynamically updating the frequently-asked-questions (FAQs), updating the knowledge repository etc. In this paper we show the utility of these discovered QA pairs as training data for a question-answering system.

Our approach for mining the case logs is based on a composite model consisting of two generative models, *viz*, hidden Markov model (HMM) and latent Dirichlet allocation (LDA) model. The LDA model explains the long-range dependencies across words due to their semantic similarity and HMM models the sequential patterns present in these case logs. Such processing results in crisp 'problem statement' segments which are indicative of the real customer concerns. Our experiments show that this approach finds crisp problem-statements in 56% of the cases and outperforms other alternate methods for segmentation such as HMM, LDA and conditional random field (CRF). After finding these crisp problem-statements, appropriate answers are looked up from an existing knowledge repository index forming candidate QA pairs. We show that considering only the problem-statement segments for which the answers can be found further improves the segmentation performance to 82%. Finally, we show that when these QA pairs are used as training data, the performance of a question-answering system can be improved significantly.

I. Introduction

Customer support center representatives or agents typically record a brief summary of their interaction in a call with a customer in the form of a case log. Agents typically hastily write down the name of the customer, the reason she called for or her concerns and if the call is resolved then a brief statement about the resolution. This information is recorded in a free-flowing text. The goal of such recording is primarily limited to manual inspection or future referencing when the same customer calls again. Agents are typically evaluated based on their average handling time (AHT) of calls and therefore these case logs are written in a hasty manner resulting into a very noisy description with several spelling and grammatical mistakes along with missing punctuations.

customer called in because her screen keeps blnking on and off when pop ups come on customer was on the device and i did provide me with a call back number gave the customer a call back did a master rest to see if that fixes the issue i did educate she may have to take the devcie inside the store if not resolved she didnt want to do the hardreset she said she would go inside the store
was not able to hear and was having dropped calls called in was unable to access internet and was slow did check neb did not have another phone edu. refreshed device. updated data profile and reset network settings
cus is calling in not able to receive or send out any sms messages.....had check icare and had a sms block on the phone and removed it and ##update on his phone and oma went all green and tested it to and from his phone and he got mine and i got his....issue resolved

Fig. 1. Example Case Logs

Figure I shows some example case logs. These examples show that there is a lot of valuable information in the case logs. They represent real everyday concerns being faced by the end-customers which is an extremely useful information for enterprises. Despite this, the case logs are rarely used for any other purpose other than manually looking up when the same customer calls again. This is primarily because of the free flowing noisy text where greetings, problems, resolutions are all intermingled making it hard to use them for any other purpose.

It is easy to see that if properly processed, these logs can provide information about frequently faced customer concerns, most current customer concerns, type or categories of customer concerns, concerns and their resolutions and many other useful insights. It is also possible to learn various ways (paraphrasing) of expressing the same end-customer concerns such as *'cannot send or receive sms'* and *'unable to send receive text mes-*

sages'. In this paper, we focus on processing these case logs to identify clear description of problems and other segments.

The case logs can also be mapped to existing knowledge articles for that enterprise. Knowledge articles are typically HTML pages published on the Web by the enterprise to help end-customers or to help agents in assisting end-customers. A collection of such articles is referred here as knowledge repository. The mapping of a case log to an article in the knowledge repository results in a question-answer (QA) pair. These QA pairs can be used for a variety of applications such as dynamically generating frequently-asked-questions (FAQs), finding the coverage of current knowledge repository and updating it. In this paper, we consider an important utility of these QA pairs as a set of training points for training a QA system which ranks candidate answers based on certain characteristics or features [1].

We first segment the free flowing text of case document in terms of 'problem statement' and other 'process details'. The problem-statement is the segment which crisply defines the problem being faced by the customer for which she called. In the examples given in Figure I, the problem statements are *"screen keeps blnking on and off when pop ups come on"*, *"unable to access internet and was slow"*, and *"not able to receive or send out any sms messages"* respectively. The process-details, on the other hand includes other background details such as whether the call dropped or whether the customer was angry etc. and also the diagnosis and solution of the problem.

In this paper, we first explain our approach for segmenting these documents in terms of 'problem-statement' and 'process-details' segments. Our approach is based on a composite generative model consisting of hidden Markov model (HMM) and latent Dirichlet allocation (LDA). For this purpose, we extend the approach in [2] to apply to this multi-class problem. As opposed to [2] where the objective was to generate better topic models for the semantic class by considering the syntactic dependencies between words, our objective is to also discern the problem-statement words from the process-details words. Our extended model allows us to achieve this classification as well as segmentation in one single model. Furthermore, the transitions between these two semantic classes of interests (problem-statement and process-detail) can actually model the sequence that representatives use to generate these case-logs. For example, a case document typically starts with a process-detail segment followed by a connecting word such as 'because' or 'that' followed by a problem-statement segment and our model is able to exploit and learn these patterns. This model is explained in more details in Section III.

Our composite model is a bag-of-word model where the class (or HMM state) for each word is independently sampled. This means that consecutive words can be assigned to different classes. On the other hand, we would like to form a problem statement not just based on content keywords (e.g. *screen-blnking-on-off-pop-ups-come*, *'unable-access-internet-slow'* and *'not-able-receive-send-out-sms-messages'*) but also functional words that connect these content keywords to form a more human readable problem statement. We perform a Viterbi decoding on the text using a minimum duration HMM topology to enforce these continuity constraints. The Viterbi decoding segments the entire case text in terms of problem-statement and process-details segments. This is explained in more details in Section III-A.

After the segmentation, we use the resulting problem-statement segments to form QA pairs. We exploit an existing knowledge repository published by that enterprise. Each segment is used as a query to this repository and the top retrieved result is assumed to be the answer. We present a scoring mechanism to rank these QA pairs. In fact, other than forming QA pairs, this mapping step also serves as a mechanism for judging the correctness of the problem-statement segments. Experiments shown in Section V show that the percentage of correct segments in top 2000 candidates sorted by the QA pair score is 82%, which is much higher than an average correct segment percentage of 56%.

To show a concrete usefulness of these resulting QA pairs, we use these pairs to train a question-answering system in ranking the answers to a given question. This is explained in Section V-D. We show that the system performance can be improved significantly when these QA pairs are used for training.

Overall, in this paper, we make following contribution:

1) Extend the HMM-LDA framework original proposed in [2] to support multiple content or semantic classes. This, combined with the minimum duration Viterbi decoding provides a general framework for text classification and segmentation.
2) We show that mapping the resulting problem-statement segments to an existig knowledge repository provides a good set of QA pairs and also provides more than 80% correct problem statements.
3) As an application, we show that these QA pairs can be used to train a question-answering system and this improves the system accuracy from 23% and 57% to 66% and 80% recall at 1 and 10, respectively.

II. Related Work

Text segmentation in previous work has been primarily used to refer to the task of segmenting a text stream into topically coherent segments [3], [4], [5]. The TextTiling algorithm, introduced by Hearst [3], is a simple, domain-independent technique that assigns a score to each topic boundary candidate (inter-sentence gap) based on a cosine similarity measure between chunks of words appearing to the left and right of the candidate. Topic boundaries are placed at the locations of valleys in this measure, and are then adjusted to coincide with known paragraph boundaries. Passoneau and Litman [4] present an algorithm for identifying topic boundaries that uses decision trees to combine multiple linguistic features extracted from corpora of spoken text. These include prosodic features such as pause duration, lexical features such as the presence of certain cue phrases near boundary candidates, and deeper semantic questions such as whether two noun phrases

on opposite sides of a boundary candidate co-refer. Authors in [5] used latent semantic analysis (LSA [6]) for finding topic boundaries, where they used LSA for computing inter-sentence similarity and showed that it is more accurate than the cosine metric as used in [3].

Such topic change detection approaches based on lexical coherence are not likely to work for the task of segmenting noisy case documents since our target segments and documents are indeed very small and not bounded by sentence or paragraph boundaries.

On the other hand, there are sequence analysis problems in text processing which can also be seen as text segmentation tasks, as they involve segmenting a text stream in known classes of interest such as part-of-speech (POS) tagging, named entity recognition and shallow parsing. Conditional random fields (CRFs) based approaches [7], [8] typically work very well for these natural language processing (NLP) tasks. The advantage CRFs offer over other segmentation models such as HMM is that they can make use of arbitrary contextual information (features) such as POS tags, capitalization patters, punctuation marks, word position etc. along with other lexical information. Although we have also used CRF as one of the alternate methods and compared it with our approach, the disadvantage of CRFs for the current problem is that our data is noisy with a lot of spelling mistakes, lacks punctuation and other informative cues which are available for the NLP tasks such as parsing and POS tagging. Furthermore, they require a labeled training dataset for training, which is going to vary from domain to domain and is hard to obtain.

A somewhat similar task is presented in [9] where a CRF based approach was presented to identify segments within call center conversations that convey caller intent. They used context information of the intent bearing segments to predict the presence or absence of intents within various segments. The context is represented through a set of phrase features that are frequently present in and around the intent bearing segments. These phrases, identified in a data-driven manner, are used along with conventional word features in a CRF based sequence labeling framework to assign intent/non-intent labels to each utterance in a conversation. However, since the conversations are already segmented in terms of caller/agent turns, the task reduces to a binary classification task.

In this work, we extend the HMM-LDA model proposed in [2] for text segmentation. The approach presented in [2] was motivated by the observation that text generation is actually a simultaneous interplay of syntactic and semantic dependencies between words and a generative model should be able to explain both these dependencies. In their setup, there is only one semantic class of content words represented by the topic model and several other function word classes such as punctuations, modal verbs, pronouns etc. They showed in the paper that by using the syntactic dependencies between these classes of words, one can derive better topics for the semantic class.

We extend this approach in [2] to consider multiple semantic classes while exploiting syntactic dependencies they share with each other as well as with other classes of function words. In this manner, our framework allows us to do classification and segmentation in one single framework. Furthermore, while the transitions between function words and semantic classes help learn better topics, the transitions between the semantic classes help us model the sequential patterns used by the representatives to write a case log, resulting in a better segmentation. The next section presents details of the segmentation technique employed for this purpose.

III. Case Document Segmentation

In this section we explain the probabilistic generative model that explains a simple stochastic procedure by which the free flowing case document text may be generated. We would like such model to explain two major characteristics of the case document data. First, it should explain (be able to exploit) the sequence in which our classes of interest (problem-statement and process-details) appear in a case and how they are interleaved via other general classes of words such as articles, modal verbs, conjunctions (e.g. although, because) etc. This can be seen as short range or syntactic dependencies across words. Second, it should explain the semantic correlations between words which is gathered from a corpus-wide agreement. For example, from a lot of context, we may want to learn that 'area', 'location' and 'home' may all point to the same semantic topic in a cellular area network domain.

We employ a hidden Markov model (HMM) to explain the short range or syntactic dependencies between various classes of words. We consider a total of eight classes: two classes of interest and 6 background classes including articles, pronouns, conjunctions, modal verbs, prepositions and punctuation marks. This will allow us to learn and enforce the sequential constraints between these classes of words. This model will allow us to learn when to emit a problem-statement or a process-details or one of the background classes words. If we decide to emit a word from say the problem-statement class then we would be required to know which word to emit. In other words, we would like to get a probability distribution over words for the classes of interest. For this purpose, we employ latent Dirichlet allocation (LDA) model.

The graphical model of our composite model is shown in Figure 2. This model is defined in terms of four sets of variables: a sequence of words $\mathbf{w} = w_1, ...w_n$ with each w_i being one of the $\mathbf{W}$ words, a sequence of class assignments $\mathbf{c} = c_1, ...c_n$ with each c_i being one of the $\mathbf{C}$ classes, a sequence of problem-statement-topic assignments $\mathbf{z_p} = z_{1,p}, ...z_{n,p}$, with each $z_{i,p}$ being one of the $\mathbf{T_p}$ problem-statement-topics and a sequence of process-details-topic assignments $\mathbf{z_o} = z_{1,o}, ...z_{n,o}$, with each $z_{i,o}$ being one of the $\mathbf{T_o}$ process-details-topics. Out of the $\mathbf{C}$ classes, the two classes of interest, say $c = 0$ and $c = 1$, are considered to be semantic classes. Since we are required to infer the topic only when the class is semantic, $\mathbf{T_c}$ is actually one for the background classes, i.e. $\mathbf{T_c} = 1$ for $c > 1$.

For each of the two semantic classes, we model the probability distribution over words as a mixture of topics.

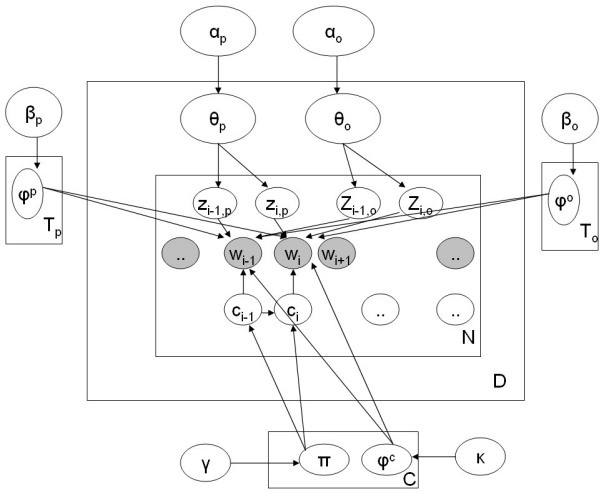

Fig. 2. The graphical representation of the composite HMM LDA model used for case document segmentation. The subscripts or superscripts p and o in the figure represent problem-statement-topics and process-details topics, respectively.

Each topic $z_{\cdot}$ is associated with a distribution over words $\phi^{z_{\cdot}}$. Each non-semantic class $c_i, i > 1$ is associated with a distribution over words ϕ^c. Each case document d has a distribution over problem-statement-topics $\theta^{d,p}$ and a distribution over process-details-topics $\theta^{d,o}$. Most importantly, transition between classes c_{i-1} and c_i follow a distribution $\pi^{c_{i-1}}$. Accordingly, the generative process runs as follows:

1) For every topic z_p and z_o,
   a) Draw Discrete distribution ϕ^{z_p} from a Dirichlet(β_p).
   b) Draw Discrete distribution ϕ^{z_o} from a Dirichlet(β_o).
2) For every class c,
   a) Draw Discrete distribution π^c from a Dirichlet(γ).
   b) Draw Discrete distribution ϕ^c from a Dirichlet(κ).
3) For every case document d,
   a) sample $\theta^{d,p}$ from a Dirichlet(α_p).
   b) sample $\theta^{d,o}$ from a Dirichlet(α_o)
   c) For every word w_j in document d,
      i) Sample class, c_j, from Multinomial($\pi^{c_{j-1}}$)
      ii) Sample problem-statement-topic $z_{j,p}$ from Multinomial($\theta^{d,p}$)
      iii) Sample process-detail-topic $z_{j,o}$ from Multinomial($\theta^{d,o}$)
      iv) if $c_j = 0$ then sample w_j from $\phi^{z_{j,p}}$, else if $c_j = 1$ then sample w_j from $\phi^{z_{j,o}}$, else sample w_j from ϕ^{c_j}

For inferencing, we use Gibbs sampling [10] to sample class and topic for a word from a posterior distribution over assignment of words to classes and topics. We assume that the document specific distributions ($\theta^{d,p}$ and $\theta^{d,o}$) over problem-statement-topics and process-details topics are drawn from symmetric Dirichlet(α) distributions, the topic distributions (ϕ^z) are drawn from symmetric Dirichlet(β) distribution, the rows for the transition matrix of the HMM are drawn from a Dirichlet(γ) distribution and class distribution over words ϕ^c are drawn from Dirichlet(κ) distribution.

Given the words $\mathbf{w}$, class assignments $\mathbf{c}$, other topic assignments $.\mathbf{z_{-j}}$, the problem-statement topic $z_{j,p}$ and process-details-topic $z_{j,o}$ for word w_j are drawn as follows:

$$P(z_{j,p}|\mathbf{z_{p}}_{-j}, \mathbf{c}, \mathbf{w}) \propto P(z_j|\mathbf{z_p}_{-j}) \; P(w_j|\mathbf{z_p}, \mathbf{c}, \mathbf{w}_{-j}) \begin{cases} n^{d_j}_{z_{j,p}} + \alpha_p & \frac{n^{z_{j,p}}_{w_j} + \beta_p}{n^{z_{j,p}}_{\cdot} + \mathbf{W} * \beta_{\mathbf{p}}} \quad c_j = 0 \\ n^{d_j}_{z_{j,p}} + \alpha_p & \quad c_j \neq 0 \end{cases} \tag{1}$$

where n^{d_j} is the number of words in the document d_j where the word w_j appears, n^z_w is the number of times word w is assigned to topic z. The process-details-topics $z_{j,o}$ for each word w_j are drawn similarly as above while making a special case for $c_j = 1$.

Similarly, class c_j for each word w_j is sampled from conditional distribution as follows:

$$P(c_j|\mathbf{c}_{-j}, \mathbf{z_p}, \mathbf{z_o}, \mathbf{w}) \propto P(w_j|\mathbf{c}, \mathbf{z}, \mathbf{w}_{-j}) \; P(c_j|\mathbf{c}_{-j}) \begin{cases} \frac{n^{c_j}_{w_j} + \kappa}{n^{c_j}_{\cdot} + \mathbf{W}\kappa} & P(c_j|\mathbf{c}_{-j}) \quad c_j > 1 \\ \frac{n^{z_{j,p}}_{w_j} + \beta_p}{n^{z_{j,p}}_{\cdot} + \mathbf{W}\beta_p} & P(c_j|\mathbf{c}_{-j}) \quad c_j = 0 \\ \frac{n^{z_{j,o}}_{w_j} + \beta_o}{n^{z_{j,o}}_{\cdot} + \mathbf{W}\beta_o} & P(c_j|\mathbf{c}_{-j}) \quad c_j = 1 \end{cases} \tag{2}$$

$$\tag{3}$$

where $P(c_j|\mathbf{c}_{-j})$ is given by:

$$P(c_j|\mathbf{c}_{-j}) = \frac{(n_{c_j}^{c_{j-1}} + \gamma)(n_{c_{j+1}}^{c_j} + I(c_{j-1} = c_j).I(c_j = c_{j+1}) + \gamma)}{(n_{c_j}^{c_{j-1}} + \gamma)(n_{\cdot}^{c_j} + I(c_{j-1} = c_j) + \mathbf{C}\gamma)} \tag{4}$$

Note that if $\mathbf{T_p}$ and $\mathbf{T_o}$ are set to 1, then this model will essentially reduce to a simple HMM where each state (class in our case) will emit words based on the distribution of words for that class and where that distribution is based on a single topic.

The model inferencing starts from a random initialization to begin with. However, since there are too many variables to learn, we provide guided initialization for the known grammatical classes such as articles, pronouns, conjunctions etc. This means that if w_j belongs to one of these classes then that word is automatically assigned that class and no sampling is performed. Furthermore, we found that obtaining a set of domain terms in any domain is not that difficult if a set of technical or knowledge articles are present. We run a POS tagger on the text of these articles to find all the terms tagged as nouns. After computing the frequency and removing the spurious terms, we keep a total of approximately 1400 domain terms. These terms are assigned class $c = 0$ from the beginning and the class for these words is not sampled from iteration to iteration. However, the topics for these words are still sampled and will eventually affect the word probability distribution for the HMM states.

Table I shows the top 15 words for each of the topic-word distributions ϕ^{z_p} and ϕ^{z_o} where we set $\mathbf{T_p} = 6$ and $\mathbf{T_o} = 2$. The table shows that the model is clearly able to separate the process-details words such as 'called' from the problem oriented words such as 'password'. Also, although the problem class was well initialized with good number of domain terms, the process-details topics are pretty much lernt from scratch. The table shows that the model was able to learn, for example, various (noisy) ways in which agents mention the customers. Furthermore, even among the problem topical words, many new words (e.g. 'puk' are identified) and placed in correct class and assigned to correct topic. Such new topical word assignments are not well represented in the Table though since it only highlights top 15 words.

To further understand how the HMM modeling helping in separating problem words from the process-details words, we modeled the generative process of the same dataset using only LDA. Table II shows the first 15 words for the 8 topics identified by the LDA model. This table shows that without using the sequential patterns, it is difficult to form separate topics for process-details words and problem-statement words only based on semantic context or co-occurrence.

A. Segmentation

The model explained so far considers and models each document as a bag-of-word. The classes and topics for each word are independently sampled. This means that classes (problem, process or one of the background) for consecutive words may be ever-changing and may not be consistent. In other words, as a result of the processing explained above, we would be able to get key problematic keywords (e.g. *'unable access internet slow'*) but they will not make a human readable problem-statement. Also, there are many words (e.g. *'disconnect', 'update', 'message',etc.*) which can belong to both the problem-statement class or the process-details class. For example, in the sentence, *'....cannot send or receive* **message***...'*, the term *'message'* is associated with a customer concern as opposed to the sentence, *'.....left a* **message***...issue resolved'*, where the same term is used as a process word. It is clear from these examples that in these situations it is the context which would help resolve the class assignment of the word.

To address this, we add additional constraint that any reasonable segment, be it problem-statement or process-detail, has to be of a minimum length. Such constraint would not only help human readability but it will also 'correct' some of the spurious words which could not be resolved by the above model correctly, for example words which are not very frequently occurring and may be present in both our semantic classes.

However, instead of forcing these constraints heuristically, we use Viterbi decoding [11] for this intended segmentation. The HMM topology we consider for this purpose is shown in Figure 3, where we keep only the two classes (states) of interest, problem-statement-states (S_p) and process-detail-states (S_o), each with a minimum duration of 4 words. The minimum duration is encoded by having four identical sub-states connected in strict left-to-right fashion as shown in Figure 3.

For the segmentation, we would like to get the state sequence $\mathbf{q} = \{q_1, ...q_n, q_j \in \{S_p, S_o\}\}$ for the word sequence $\mathbf{w} = \{w_1, ...w_n\}$ that maximizes the joint likelihood $P(\mathbf{w}, \mathbf{q}|\Lambda)$ as follows:

$$\mathbf{q}^* = argmax_{\mathbf{q}} P(\mathbf{w}, \mathbf{q}|\Lambda) \tag{5}$$

where Λ denotes all the parameters for the model in Figure 3. In fact, all the computations required for solving Eq. 5 are done using the parameters of our composite model in Figure 2 as explained below. Accordingly, Λ consists just of the parameters of the composite model and no additional parameters have to be lernt.

For solving Eq. 5 using the Viterbi algorithm, we use the following quantity:

$$\delta_j(i) = max_{q_1,q_2,...q_{j-1}} P(q_1, q_2, ...q_j = i, w_1, w_2, ...w_j|\Lambda)$$

i.e. $\delta_j(i)$ is the best score (highest probability) along a single path, at observation index j, which accounts for the first j observations and ends at the i^{th} state. By induction, we have:

$$\delta_{j+1}(i) = [max_k \delta_j(k) \cdot P(q_{j+1} = i|q_j = k)] \cdot P(w_{j+1}|q_{j+1} = i) \tag{6}$$

TABLE I
TOP 15 WORDS FOR VARIOUS PROBLEM AND PROCESS-DETAIL TOPICS

Problem Topics						**Process Topics**	
airave	store	service	email	internet	update	cust	yes
request	replacement	area	password	hotspot	profile	staed	xxx
forwarding	take	coverage	lost	connect	text	called	cst
bill	screen	outage	contacts	wifi	settings	informed	4
payment	battery	voice	card	connecting	test	tht	9
gps	charge	problem	app	mobile	send	bc	x9
modem	replace	roaming	gmail	spot	messages	walked	cellsite
router	freezing	home	sd	access	failed	cci	single
support	charging	open	google	auto	sms	ver	3x
list	replaced	location	puk	shooting	updated	understood	repeat
conference	asurion	market	stolen	xgb	nai	sts	qos
understand	center	following	sim	enable	mail	said	x3x
lights	insurance	experienced	sync	acess	message	custmr	worst
ending	model	evdo	user	enabled	prl	$\langle name \rangle$	xx4
person	charger	lte	download	tablet	cookies	resolved***	sector

TABLE II
TOP 15 WORDS FOR VARIOUS TOPICS FOUND BY THE LDA MODEL. IT IS SEEN HERE THAT THE PROBLEM-TOPICAL WORDS AND PROCESS-DETAIL WORDS (BOLD-FACED) ARE MIXED UP ACROSS VARIOUS CLUSTERS.

airave	reset	**cst**	**call**	**customer**	phone	device	**cust**
home	device	**xx**	**back**	his	store	**profile**	device
service	data	service	phone	account	device	update	**called**
caller	internet	**calls**	phone	email	order	refreshed	**cci**
coverage	cu	**0x**	number	sprint	customer	data	issue
cx	access	area	cus	com	battery	text	stated
issue	network	ticket	calling	phone	take	unable	issues
customer	working	voice	gave	advised	replacement	send	**edu**
sme	phone	**x0x3**	line	contacts	sprint	oma	adv
repeat	settings	issues	calls	account	rep	make	acct
time	hotspot	x0xx	steps	password	screen	test	states
roaming	connect	**dropped**	vm	wants	stated	able	informed
signal	**cci**	network	voicemail	info	told	calls	said
issue	wifi	issue	cci	lost	advised	refresh	calling
per	hard	outage	callback	card	eticket	receive	wanted

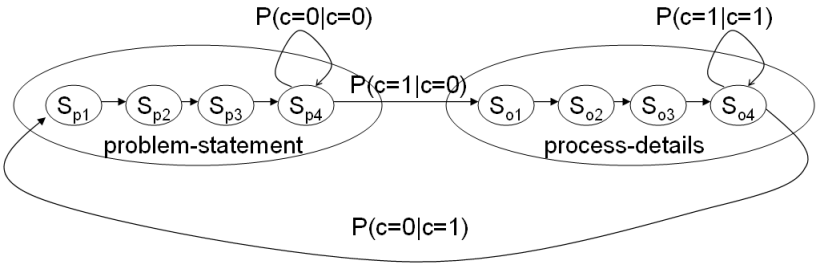

Fig. 3. The HMM topology used for Viterbi segmentation. The probabilities $P(.|.)$ and emission probability distributions are used as lernt from the composite HMM LDA model.

If we keep track of the argument which maximizes Eq. 6 for every j and i, then the most likely state sequence can be backtraced easily [11]. For computing, $P(q_{j+1} = i|q_j = k)$ in Eq. 6, we use the computation $P(c_j|c_{j-1})$ (Eq. 4) if there is a transition across S_p and S_o otherwise we consider it to be zero. For the emissions likelihoods $P(w_{j+1}|q_{j+1} = i)$, we use the following:

$$\begin{aligned} p(w_j|q_j = S_p) = & \quad \textstyle\sum_{z \in T_p} p(z|\theta_p) p(w_j|\phi^{z,p)}) & c_j = 0 \\ = & \quad 1 - p(w_j|q_j = S_o) & c_j = 1 \\ = & \quad 0.5 & c_j > 1 \end{aligned}$$

$$\begin{aligned} p(w_j|q_j = S_o) = & \quad \textstyle\sum_{z \in T_o} p(z|\theta_o) p(w_j|\phi^{z,o}) & c_j = 1 \\ = & \quad 1 - p(w_j|q_j = S_p) & c_j = 0 \\ = & \quad 0.5 & c_j > 1 \end{aligned}$$

where the class assignment c_j for each word w_j, topic-word distributions ($\phi^{z,\cdot}$) and document-topic distributions ($\theta_\cdot$) are readily used from the previous processing to compute state emission likelihoods. The minimum duration constraint as shown in Figure 2 will make sure that no segment is less than 4 words long while maximizing the objective function in Eq. 5.

The Viterbi decoding is going to result in a number of

problem-statement and process-detail segments. In general, we observed that the very first problem-statement segment was reflective of the real customer concern and the subsequent problem-statement segments were either providing more details about the problem or they were about the solution to the problem that the representative provided. Therefore, in the subsequent analysis and evaluation, we will consider only the first problem-statement segment.

After these problem-statement segments have been identified, the next step is to generate QA pairs from them. The next section provides details about our approach for generating these QA pairs.

IV. Generating QA pairs

In this paper, we consider using these problem-statement segments for generating question-answer (QA) pairs. Although such QA pairs may be required and useful for a variety of applications, here we focus on using them for training a answer ranking function for a smart question answering system [1]. Accordingly, we are interested in finding out segments which correspond to documents in the existing knowledge repository.

Typically, in a technical help scenario, there are knowledge articles intended to be used by enterprise representative answering customer concerns or the customers themselves for self-help. For example, in a smart-phone domain, the article http://support.sprint.com/support/article/Troubleshoot_issues_related_to_low_internal_memory_on_your_Samsung_Galaxy_Tab_2_101/9214bac7-6e11-47f7-afc1-5689f22d61b9 specifies steps to be taken for a specific device and for a specific problem. Similarly in a finance domain a knowledge article such as http://support.quickbooks.intuit.com/support/pages/inproducthelp/core/qb2k12/contentpackage/core/payments/task_baddebt.html would provide help for people looking to record a bad debt.

We collected approximately 4000 such knowledge articles for our experimentation. These documents were processed, cleaned, and parsed and finally indexed using Lucene [12]. Each hypothesized problem-statement segment, as found in the previous step, was used as a query to this index. For a standard querying mechanism, the retrieval score is reflective of the similarity between a document and the given query as follows:

$$sim(q,d) = \frac{V(q) \cdot V(d)}{|V(q)| \cdot |V(d)|} \tag{7}$$

where $V(q)(V(d))$ represent the vector space representation of the query q and the document d, respectively and $V(.)$ represents the Euclidean norm.

Top 3 results sorted based on the Lucene retrieval score are considered for the subsequent processing and ranking. Since it is always possible to retrieve some document or the other from the corpus in response to a query, we realized that just the score itself was not a good metric of judging the quality of the retrieval and segmentation. Particularly, we realized that the retrieved documents were trustworthy only when the score of the top document was much higher compared to the two other results. On the other hand, if the scores of all the documents were nearly the same, albeit high, these retrieved documents are not reliable. Consider for example the segment query: *'wanted to unsubscribe to visual voicemail'*. This query actually is 'transactional' in nature since it requires active agent participation with the system to resolve the customer concern about unsubscribing. The answer is not likely to be found in any of the knowledge articles. For this query, the top 3 retrieved documents are titled, *1. 'get visual voicemail to text'*, *2. 'configure visual voicemail to text'* and *3. 'reply to visual voicemail message'*. Since all of these documents are tangential to the query requirements, their scores may be high but they are all in the same range.

TABLE III
VARIOUS WAYS OF EXPRESSING THE CUSTOMER CONCERNS ABOUT *'changing caller id name'*

caller id name is incorrect
wanting to change the caller id
wants to change the caller id for this number
name showing from caller id is not showing correctly

To capture the reliability of the sorted retrieved results (d_1, d_2, d_3) given a segment query q, we compute the following reliability quotient $R(q, d_1)$:

$$R(q,d_1) = \frac{(sim(q,d_1) - sim(q,d_2)) + (sim(q,d_1) - sim(q,d_3))}{2} \tag{8}$$

When $R(q, d_1)$ is high then it provides a measure of how distinct the document d_1 is to d_2 and d_3 and this provides an indication of how truly the document d_1 matches the query q.

We consider the reliability $R(q, d_1)$ in combination with the similarity score $(sim(q, d_1))$ to score a QA pair (d, q) as:

$$Score(q,d) = R(q,d) \cdot sim(q,d) \tag{9}$$

We use the score in Eq. 9 to sort the resulting QA pairs (d, q) for their 'goodness' where q provides the 'question' and the document d serves as the answer to the question q. Evaluation presented in the next section (Tables VI and VII) is based on this sorting. The evaluation in Table VII specifically points out that as we consider more and more QA pairs in the sorted list, the average 'goodness' of the QA pairs decreases.

Note that this step does not only result in finding QA pairs but also it provides us good problem-statements to consider which can serve as likely and valid questions in this domain. We show in Section V that when sorted based on the score (Eq. 9), the percentage of correct segments improves to 82% from an average correct segment percentage of 56%.

Furthermore, such segmentation and then mapping them to existing knowledge repository also provides various paraphrasing candidates of the similar customer concerns. As an example, by the way of this mapping we found that all four problem-statements in Table III point to the same underlying customer concern since they were all mapped to the same knowledge article with a high score.

In the next section, we present our analysis of the number of correct QA pairs generated and how the accuracy of finding problem-statement segments also improves when we consider the mapping of segments to the existing knowledge repository, as discussed above.

V. Experiments and Evaluation

So far we have discussed techniques for first segmenting a given noisy case document into segments pertaining to problem-statement and process-details segments. Further, we have discussed mapping these documents to existing knowledge repository so as to generate QA pairs. In this section, we present evaluation of these techniques against other possible alternate methods. We first present the details of the dataset that we used to conduct these experiments and evaluations.

A. Dataset

We used a total of 65000 case documents that corresponded to approximately one month of customer-agent conversations in a smart-phone customer support environment. The average length of these documents in terms of words is approximately 45 words with a standard deviation of approx. 30 words. We picked subsets of this data set for evaluating various components as discussed in the following.

B. Segmentation Accuracy

One of the key tasks presented in this paper relates to automatic segmentation of case documents. We first present the evaluation of our approach on this key task and compare it with other possible solutions. For this study, we used 500 randomly picked case documents and their segmentation. For each of these documents, we had a human annotator label the resulting problem-statement segment as correct or incorrect based on whether the segment was informative by itself as a problem-statement. As an example, segments like *'google play store is not working'* were marked correct whereas segments such as *'line ending xxx3 stating'* were marked incorrect.

Table IV presents the comparison. These comparisons show that our approach performs best among the alternatives. The HMM-only model, as mentioned earlier, results when we have $\mathbf{T_p} = \mathbf{T_o} = 1$ set in our composite model while keeping everything else the same. This way, each class or HMM state is modeled with just one topic or vocabulary. It is not surprising to see that this performs worse than the proposed approach since a single vocabulary code-book is not capable of aggregating semantically similar words together as done by mixture of topic models (LDA).

Conditional random fields (CRFs) [13] are a type of discriminative undirected probabilistic graphical models. CRFs have long been successfully used for text processing tasks such as segmentation, parsing [8], labeling [14] and named-entity recognition [7]. They provide a better alternative to HMMs, especially for text processing tasks because they can exploit the 'context' of observations without assuming them independent. CRFs are trained in a supervised manner using a labeled dataset. Although our proposed composit model is completely unsupervised and comparing the performance with a completely supervised method would be unfair, we wanted to evaluate how much comparable or better segmentation can be provided by CRFs. We used the Mallet CRF implementation [15] for this purpose.

TABLE IV
Comparison of the proposed composite model with HMM and CRF models

Method	Segmentation Accuracy
Proposed composite model	56%
HMM only	48%
CRF	41%

We designed the CRF training and testing as follows. First, we manually created problem-statement and process-details segments corresponding to 500 case documents. Next, to keep enough context in each observation and to avoid sudden change in the observation label, we generated windows of five words each to constitute an observation. We used a sliding window mechanism such that two consecutive observations differed only in one word to ensure that the observation labels (expected output label during testing) were smooth. Such mechanism also allowed us to evaluate the CRF output on a per-word basis. This resulted in a total of 5442 segments, 4196 problem-statements and 1246 process-details segments. A held-out set of 500 case documents was separately arranged as observations in exactly the same manner as training data i.e. observations of 5 words with one word being different in two consecutive observations. Furthermore, we found that CRF performed better when each domain term, as used in our composite model, was marked consistently as 'DT'. This enabled the model to learn a segment as 'problem-statement' when a lot of domain specific terms are present in it.

The Table IV shows that our method which is completely unsupervised also outperforms a supervised CRF model. This can be explained by the fact that whereas CRF is good at learning class-to-word discrimination and also good at learning appropriate transitions among classes but it fails at learning semantic context or similarity of different words. Since the data being used for these experiments is highly noisy with lot of spelling mistakes, a feature representing strict class-to-word association would tend to fail. LDA, on the other hand, can learn the semantic topic of these erroneous words based on its cleaner context and may establish the correct class-to-word association. We also found that the CRF model failed to generate any problem-statement segment on very small case documents (last two rows in Table V). In fact, out of a total of 500 case documents, problem-statement segments were found by CRF in only 80% of the documents of which 41% were correct as shown in Table IV.

C. Question-Answer Pair Generation Evaluation

As explained in Section IV, we map the resulting problem-statement segments to existing knowledge repository articles to form QA Pairs. This, in turn, also helps us in automatically removing some of the bad segments. Table VI shows that the

TABLE V
EXAMPLES OF PROBLEM-STATEMENT SEGMENTS FOUND BY THE THREE APPROACHES

Input Case Document	Proposed Composite Model	HMM-Only model	CRF
name: YYYY auth meth: pin reason/request: cci issue accessing the intenrte samsung phone no alternate # number: xxx-xxx-xxxx info/instruc : checked for outage / advised customer to turn off the phone remove the battery/refreshed the device/ gave customer instrucation to update the profile/ actions/results: setup followup xxx-xxx-xxxx	issue accessing the intenrte samsung phone	pin reason / request : cci issue accessing the intenrte samsung phone / no alternate # number	request cci issue accessing the intenrte samsung phone no alternate number xxx-xxx-xxxx info instruc
ptn:xxxxxxxxxx name:YYYY auth:pin issue:dropp calls and cannot speak to anyone without being on speakerphone info:schedueling callback with customer tomorrow x0am to perform troubleshooting result:resolved rpt caller:no	issue : dropp calls and cannot speak to anyone without being on speakerphone info	issue : dropp calls and cannot speak to anyone without being on speakerphone info	issue : dropp calls and cannot speak to anyone without being on speakerphone info schedueling callback with customer
custhas no data connectivity// checked for outages found evdo connection xxxxxxx xx xx xxxx 0x:xx:00 cst//edu cust about outage //checked to make sure mobile data is enabled ##scrtn#//resolved	custhas no data connectivity	outages found evdo connection	custhas no data connectivity checked for outages found evdo connection xxxxxxx xx xx xxxx
issue: the device is stolen : xxxxxxxxxxxxxxxxx customer swapped to her old evo the agent before me failed to inform her of the claim process therefore she swapped first before filing actions	issue: the device is stolen	issue: the device is stolen	issue the device is stolen xxxxxxxxxx0xxxxxxx customer swapped to
cci because she can not access internet	not access internet	not access internet	[NOT FOUND]
unable to get on craigs list. also has some network issues.	unable to get on craigs list. also has some network issues.	unable to get on craigs list. also has some network issues.	[NOT FOUND]

TABLE VI
COMPARISON OF THE PROPOSED COMPOSITE MODEL AFTER POST PROCESSING WITH HMM AND CRF MODELS

Method	Segmentation Accuracy
Proposed composite model	56%
Proposed composite model after mapping	**82%**
HMM only	48%
CRF	41%

segmentation accuracy as a result improves from 56% to 84%.

We also compute the percentage of QA pairs which are correct. To compute this we take the top 1000 and 2000 QA Pairs ranked based on the score computed in Section IV. As expected, we find that when the segment quality is bad, the resulting answers are also bad since it is difficult to retrieve a good answer from the index if there are more noise-words (process-details words) than the problem-statement words. On the other hand, when a correct problem-statement segment is used to query the index, 64% of the times it results in correct answer as well as shown in Table VII. On the remaining 36% occassions, a correct answer could not be found for reasons such as:

- The segment involved a transaction for which there is no knowledge article. These must be dealt with seperately. For example *'he order device and never receive'*.
- The correct answer may have been lower down the ranked result list. Since we are considering only the top result, it will be missed. For example, for a query *'forward calls to another number'*, the correct result title is *'Call Forwarding'* which has less word match compared to the top result *' forward voicemail message to another number'*.
- In some cases the resulting segment is good in that it was the best problem-statement possible from a given case document. Consider for example the case document: *'cust ci to get help with bad data cust is on device no call back educated cust offline troubleshooting problem solved'*. The segment extracted using the proposed approach was *'get help with bad data'* which is the best possible segment that could have been extracted. However, this does not result in correct documents such as *'bad data connectivity issues'* since terms like 'connectivity' and 'issues' are missing from the segments as well as the original case text.

Table VII also shows that as we consider more and more (1000 and then 2000) QA Pairs sorted based on the scores as explained in Section IV, the percentage of correct segments drops from 84% to 82% (the sum of the first two rows in Table VII) and the percentage of correct QA pairs drops from 53% to 48%. This suggests that the score (Eq. 9) as explained in Section IV provides us a good measure of mapping between problem-statement segments and the indexed corpus.

Next, we present an analysis of how these good QA pairs can affect the performance of a question-answering system.

D. Application of automatically generated QA Pairs

The QA pairs generated using our approach can be used for various purposes such as:

1) Generating frequently asked questions (FAQs) using most recent case logs
2) Finding various forms (paraphrasing) used by customers to report their concerns

TABLE VII
PERCENTAGE OF CORRECT QA PAIRS GENERATED AUTOMATICALLY

Segment Quality	QA Pair Quality	% of top 1000 documents	% of documents top 2000 documents
good	good	53%	48%
good	bad	31%	34%
bad	good	3%	3%
bad	bad	13%	15%

3) Automatically finding the coverage of the current knowledge repository
4) Topical analysis of the problems being faced by the customers using the topics discovered by the composite model.
5) Training a question answering system

Each of these applications would require an appropriate processing and consumption of the resulting QA pairs. Here we focus on the application of using these QA pairs for training a question answering system. Figure 4 shows architecture of the question answering system that we used for this analysis.

Although, the details of the question answering system illustrated by the architecture in Figure 4 are beyond the scope of this paper, we discuss the components in brief for clarity. The architecture shown in Figure 4 suggests a typical pipeline flow of components required for question answering. The question analysis component in the figure refers to the activities related to understanding and parsing the user query, determining and highlighting keywords, finding key relations shared by the words in the query etc. The query formulation component then builds upon the question analysis findings and then expands the query using synonym information and proximity information resulting from the relations discovered earlier. Answer indices refer to the knowledge repository that we have used earlier in Section IV. Candidate answers are the documents returned as the result of querying the answer indices using the queries formed in the query formulation component. For each candidate answer thus generated, a set of 'features' are generated and referred to as answer scorers. Examples of such features include the number of common word relations shared between question and answer, the retrieval score and ranks resulting from querying the answer indices, similarity of answer titles with the question, textual alignment similarity between the title and the question etc. There are 22 features used in the instance of the system that was used for the following evaluation. Finally, the answers are ranked by evaluating these features via a trained model. The model used in the instance used for this study was logistic regression.

It is clear from this description that one of the key components in such system is the trained model. The training of such model requires labeled data where correct or true answers are marked for a number of questions. Assuming rest of the answers as incorrect, the model essentially learns the parameters (weights of each of the features) which are most helpful in bringing the correct answers at the top of the result list.

Generating such labeled data manually is a tedious and time consuming task. It first requires a list of possible questions which can be used by labelers to mark correct answers. For each such question candidate, the labelers have to be provided a set of possible answers to choose the correct answers from. There are other considerations that one has to follow such as:

1) Consider questions from a wide variety. In a financial domain, for example, these varieties would include reporting, printing, billing, listing,data etc. In a smartphone domain, these varieties should include network, applications, voice quality etc.
2) Consider different forms of framing a question. This is required since one of the aims of the training exercise is also to enable the model to learn the effect of these different forms and what features can be instrumental in bridging the gap between how the question is asked and how the answer is stored in the indices.

Considering that not all question candidates may result in a suitable answers combined with above consideration provide an idea of how tedious this exercise can become. One good advantage of the proposed approach is that it implicitly addresses the two major considerations mentioned above. Since we are getting the data from case logs, we are likely to find the paraphrasing of similar concerns automatically. Also, since case logs are recorded for all variety of concerns, we are likely to have samples of all type of questions that can be asked in that domain. Even better, we are likely to get them in the similar proportion of frequency as they appear in the everyday call logging.

In the following we present our analysis of how our resulting QA pairs affect the training of the question answering system. These evaluations were done on a set of 100 held out questions (test-set). Figure 5 shows the performance of the question answering system as a function of how many automatically generated QA pairs were used for training in addition to 200 manually labeled QA pairs. As a baseline, we present evaluation results without using any of the automatically generated QA pairs and while the model was trained using only 200 manually labeled QA pairs.

Figure 5 shows that using the automatically generated QA pairs significantly improves the performance of the system. The performance of the system in Figure 5 is shown in terms of $Recall@N$ which measures how many questions found their correct answers when ranks up to N are considered. The figure also shows that using more and more QA pairs, while keeping rest of the system configuration exactly the same, further improves the performance.

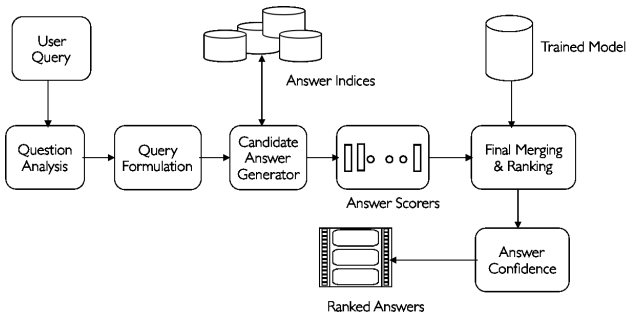

Fig. 4. Architecture of a question answering system used for evaluating the effect of automatically generated QA Pairs

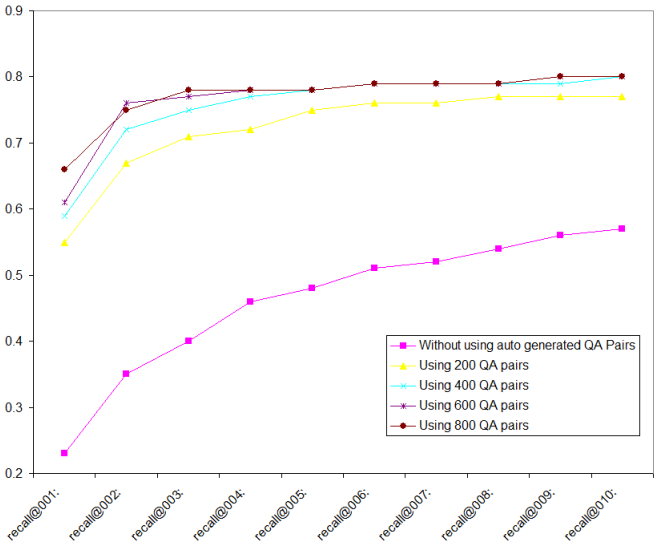

Fig. 5. Comparing system performance without and with automatically generated QA pairs. The figure shows that the system accuracy improves highly when automatically generated QA pairs are used for model training

VI. Conclusion

This paper presented a technique for segmenting noisy case logs in terms of problem-statement and process-details segments where the problem-statements reflect the real customer concerns whereas the process-details capture all other details such as greetings, details of the customer and diagnosis of the problem etc..

We use these problem-statement segments to generate question-answer (QA) pairs in a given domain by mapping them to a existing knowledge articles in that domain. We also propose a scoring mechanism to rank these QA pairs that takes care of transactional or other unanswerable problem-statement segments. When sorted based on this score, we show that as many as 50% of such automatically generated QA pairs are good and can be readily used for a target application such as training a question-answering system. In addition to QA pair generation, we show that after this mapping and scoring and considering top 2000 candidates, the percentage of correct problem-statement segments improves to 82% as compared to an average correct percentage of 56%. Finally, we show that when used as training data for a question-answering system, these generated QA pairs improve the performance of the system significantly from 23% and 57% (Recalls at 1 and 10) to 66% and 80%.

Such generated QA pairs and problem-statement segments can be used for a variety of application, especially in a customer support ecosystem. Other than the improved question-answering system, these QA pairs and problem-statements can be used for analysis such as dynamically generating FAQs, finding various forms of expressing similar customer concerns (paraphrasing) and accordingly adapting the information storage and retrieval systems.

References

[1] D. A. Ferrucci, "Introduction to "This is Watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, May 2012.

[2] T. L. Griffiths, M. Steyvers, D. M. Blei, and J. B. Tenenbaum, "Integrating topics and syntax," in *In Advances in Neural Information Processing Systems 17*. MIT Press, 2005, pp. 537–544.

[3] M. A. Hearst, "Multi-paragraph segmentation of expository text," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, pp. 9–16. [Online]. Available: http://dx.doi.org/10.3115/981732.981734

[4] R. J. Passonneau and D. J. Litman, "Discourse segmentation by human and automated means," *Comput. Linguist.*, vol. 23, no. 1, pp. 103–139, Mar. 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=972684.972689

[5] F. Y. Y. Choi, P. Wiemer-Hastings, and J. Moore, "Latent semantic analysis for text segmentation," in *In Proceedings of EMNLP*, 2001, pp. 109–117.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[7] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 188–191. [Online]. Available: http://dx.doi.org/10.3115/1119176.1119206

[8] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 134–141. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073473

[9] S. Ikbal, A. Verma, P. Ghosh, K. Church, and J. Marcus, "Intent focused summarization of caller-agent conversation," in *ICASSP*, 2013.

[10] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning." *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/ml/ml50.html#AndrieuFDJ03

[11] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.

[12] M. McCandless, E. Hatcher, and O. Gospodneti, "Lucene in action," 2008.

[13] J. Lafferty, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[14] Q. Huang, M. Han, B. Wu, and S. Ioffe, "A hierarchical conditional random field model for labeling and images of street scenes," in *International Conference on Computer Vision and Pattern Recognition*, 2011.

[15] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.