Exploiting a Proximity-based Positional Model to Improve the Quality of Information Extraction by Text Segmentation

Dat T. Huynh **Xiaofang Zhou**

School of Information Technology and Electrical Engineering
The University of Queensland, Australia
Email: {tandat.huynh, zxf}@uq.edu.au

Abstract

A large number of web pages contain information of entities in a form of lists of field values. Those implicit semi-structured records are often available in textual sources on the web such as advertisings of products, postal addresses, bibliographic information, etc. Harvesting information of those entities from such lists of field values is challenge task because the lists are manually generated, not written in a well-defined templates or may miss some information. In this paper, we introduce a proximity-based positional model (PPM) to improve the quality of extracting information by text segmentation. Our proposed model offers improvements over the fixed-positional model proposed in ONDUX, a current state-of-art method for information extraction by text segmentation (IETS) to revise the labels of text segments in an input list of field values. Different from fixed-positional model in previous work, the key idea of PPM is to define proximity heuristic for labels in an input list in a unified language model. Our proposed model is estimated based on propagated counts of labels through a proximity-based density function. We propose and study several density functions and experimental results on different domains show that PPM is effective to revise labels and helps to improve performance of current state-of-art method.

Keywords: Proximity, positional model, information extraction, text segmentation.

1 Introduction

Entity extraction, a typical task in information extraction, is the process of extracting entities such as people, organisations, or locations on web pages. It has become an active and hot research topic over past decade. According to the study and analysis of Guo et al. (2009), named entities occurs in about 71% of search queries of users. Nevertheless, current search engines such as Google[1] or Bing[2], which support users to search information on the web according to their queries, are mainly based on keyword or text matching techniques and do not capture the semantic information of objects and relations between objects. Therefore, the problem of entity extraction, which includes identifying named entities, their attributes as well as the relations between entities, is an indispensable task not only in query answering but also in knowledge discovery from web environment.

In our work, we focus on the problem of entity extraction in which information of entities are described in a list of field values on web pages. As some examples of the task, we consider an advertising about a car and an address of a person on web pages: *"Ford Falcon, White colour, December 1996, $2,900.00, Western Australia 6155, 0402-744-126"* or *"Dr. Janelle A. Briggs, 208 Carmody Rd, St. Lucia QLD 4067"*. Such kinds of information are often available in several textual sources on the web, such as bibliographic information, postal addresses, advertisings, recipes, etc. Therefore, it is an important practical problem of information extraction that has been frequently addressed in the recent literature. In the second example, the following structured record could be extracted:

<*name, "Dr Janelle A. Briggs"*>
<*street, "208 Carmody Rd"*>
<*area, "St Lucia"*>
<*state, "QLD"*>
<*code, "4067"*>

In the literature, the problem of entity extraction from lists of field values is addressed as the problem of information extraction by text segmentation (IETS) in which information of entities organised in implicit semi-structured records. There are various possible segmentation schemes to choose from list-specific wrappers or statistical segmentation models. Since the field values in the implicit semi-structured records are not machine-generated and they are represented in a textual representation, traditional wrapper-based methods (Crescenzi et al. 2001, Arasu & Garcia-Molina 2003) cannot be applied for the inputs which are formatted differently in HTML. A dominant approach for this problem is the deployment of statistical methods, such as Hidden Markov Model (HMM) (Borkar et al. 2001) and Conditional Random Fields (CRFs) (Lafferty et al. 2001) to extract information. In these statistical methods, an extraction model is trained based on a training dataset which consists of a set of text segments and their labels. CRFs-based methods were proven to outperform HMM-based methods and have been widely used in several information extraction systems (Sarawagi 2008, Zhao et al. 2008). They are more accurate and robust for extracting information of entities from such records because they can exploit an arbitrary number of rich and correlated properties of words in sentences. However, obtaining a large amount of training data, which includes the association between string segments with their corresponding attributes, to build an extraction model may be very expensive or even

[1] http://www.google.com
[2] www.bing.com

unfeasible in some situations. Therefore, some recent studies proposed the usage of pre-existing datasets to alleviate the need for manually labeled training data (Agichtein & Ganti 2004, Mansuri & Sarawagi 2006, Zhao et al. 2008). In these methods, known values in a database are used to train a statistical model for recognising the values of their attributes in an input text. However, they made strict assumptions about the single total order of field values in text sequences. Recently, Cortez et al. (2010) have proposed ONDUX, an unsupervised method to overcome those drawbacks. Instead of learning an extraction model from training dataset, they exploited the values of labels in a knowledge base to associate text blocks or segments in an input text with the labels by employing matching functions on the overlapping tokens between segments and knowledge base. Then, the mismatched and unmatched labels are rectified by using a sequential model and a positional model in a final "reinforcement" phase. Both sequential model and positional model are built from the input text to verify and potentially correct the assignments of labels to text segments.

Our proposed method is an improvement the study of Cortez et al. (2010). In matching phase, while the study of Cortez et al. (2010) focused on the content-related feature to define matching functions between text segments and labels, we incorporate format-related feature and combine with content-related feature to improve the performance of matching phase and hence we can obtain high performance in a final "reinforcement" phase. To combine format-related feature and content-related feature into a single similarity framework, we view each label or entity type as a set and each text segment as a member of a set in set theory and exploit the intensional and extensional definition of a set to define a membership relation between a member and its set in a formal way. Due to this, we incorporate both format-related and content-matching functions in a novel type-based similarity framework between a text segment and a label.

Moreover, positional model in reinforcement phase in previous work only considers fixed positions of a label in different lines in an input list when it revise the labels of text segments. However, in practice it cannot be ensured that a correct label always occur in a fixed position in different lines. Therefore, we relax this constraint by proposing a *proximity-based positional model* (PPM) for labels.

The key idea of PPM is to define a language model for each position of a label in an input text. The PPM for a label at a position would be estimated based on the propagated counts of the label in different positions in the input text. The occurrences of a label t at the positions which near a position i will provide more evidence than its occurrences in far positions. In other words, each position of a label will receive propagated counts of the label in near positions. A main technical challenge in the proximity-based positional model for labels in an input list is how to define a propagation function and estimate a positional model for the labels accordingly. We analyse several functions in this paper and we show that with some specific choices, our proximity-based positional model for labels covers the fixed-positional model proposed in (Cortez et al. 2010) as a special case.

To summarise, we believe that we make the following contributions.

- We propose a type-based similarity framework to asset how likely a segment string should be a member of a given set. By this way, we incorporate both format-related and content-related features in a framework to recognise labels for text segments with high performance.
- We propose a novel proximity-based positional model (PPM) for labels to relax a rigid constraint and improve performance of previous work. Instead of considering a label in a fixed position in the input text, the distribution of the target label in different positions is taken in our model to measure how likely a label occurs at a given position. Our proposed model is proven more flexible and robust than the fixed-positional model in the previous work.
- We have conducted experiments on public datasets and the experimental results prove that our proposed techniques helps to improve the performance as compared to the current state-of-art study on the problem of information extraction by text segmentation.

The remaining sections of this paper are organised as follows. Section 2 presents existing related studies on the problem of IETS in the literature. Next, we introduce and formally define our proposed proximity-based positional model (PPM) for labels in section 3. Then, we present a type-based similarity measure in matching phase and the usage of PPM in refinement phase to solve the problem of IETS in section 4. After that, our experiments and evaluations are described and analysed in section 5. Eventually, section 6 concludes the paper and suggests some future work.

2 Related work

Information extraction by text segmentation (IETS) is the process of converting an unstructured document which contains implicit records into structured form by splitting the document into substrings which contain data values (Sarawagi 2008). In other words, each text input or document forms one or several implicit records and each implicit record is represented in a form of a list of field values. The dominant approach to segment texts in an input list to extract field values in the literature is the application of machine learning techniques with two different techniques for generating training data. The first technique, which is called supervised approach, builds a training data set manually by human (Seymore et al. 1999, Freitag & McCallum 2000, Borkar et al. 2001, Lafferty et al. 2001, Peng & McCallum 2006, Mansuri & Sarawagi 2006). Meanwhile, the second technique exploits existing data in a knowledge base or reference table to build training data automatically (Agichtein & Ganti 2004, Mansuri & Sarawagi 2006, Zhao et al. 2008, Cortez et al. 2010)).

The studies of Seymore et al. (1999) and Freitag & McCallum (2000) can be considered as the first studies addressing this problem in the literature. In their work, a Hidden Makov Model (HMM) for recognising the field values in an input text was constructed from a provided training dataset. Later, this approach was extended in the system DATAMOLD (Borkar et al. 2001), in which each state of an external HMM modelling the sequence of field values in an input text contains an internal HMM. Each internal HMM is built as a model for recognising the value of each attribute. In their work, both internal and external HMM are trained from a hand-labelled dataset.

After that, Conditional Random Fields (CRFs) (Lafferty et al. 2001) was proposed as an alternative model for HMM to solve the task of labelling texts.

CRFs-based methods become popular in the field of information extraction because of their high reflexibility and good extraction results. They are proven to outperform all previous learning-based methods in both theory and experimental evaluations for the problem of sequence labelling (Peng & McCallum 2006, Sarawagi 2008). However, although the quality of extraction results of HMM and CRFs are good, these supervised methods require to have a large amount of manually training data to build their extraction model.

To reduce the dependency on manually labeled data, as proposed in the study of Mansuri & Sarawagi (2006), the authors exploited an existing structured database to define new database-related features for CRFs. Their method used only few labeled training instances to train a CRFs-based extraction model. Although this method exploited additional features from a structured database, it still needs some user-provided training data in their method.

Different from above methods, the general idea of unsupervised approach is to exploit a pre-existing datasource to build a training data for a statistical extraction model. The study of Agichtein & Ganti (2004) followed this idea and proposed an unsupervised method with HMM. The method assumes that all field values in text sequences share the same total order. The technique firstly trains an HMM model for each attribute by employing the reference table data. These trained HMM models are used to find the best start positions for every attribute in every input sequence. Then it uses these positions to infer the total order. Finally, a training dataset is directly constructed from a reference table by concatenating attribute instances in the table according to that total order.

A similar technique was proposed in the study of Zhao et al. (2008). However, the authors adapted the algorithm of Agichtein & Ganti (2004) to CRFs instead of using HMM. Different from HMM, CRFs is not a generative model and it does not model the distributions of observations. Therefore, they proposed a technique to infer the total order by augmenting negative labels and negatively labeled examples in the training data of attribute-CRFs. Due to this, the attribute-CRFs will assign low likelihood scores to incorrect starting positions of an attribute in an input sequence.

Although both two methods of Agichtein & Ganti (2004) and Zhao et al. (2008) do not require training data, it has some limitations. Firstly, both two method makes a strict assumption about the single total order of field values in text sequences. In practice, it is common to have more than one order of attributes in a single dataset. Moreover, this method has bad performance in running time because it requires to execute inference step and training step for each time it performs a new extraction on an input text.

Recently, Cortez et al. (2010) proposed ONDUX, an on-demand unsupervised approach for the problem of information extraction by text segmentation (IETS). In their method, a knowledge base is employed to label text segments in citation strings via attribute matching functions. Then, the labels of those segments are revised by a reinforcement step which uses both sequential and positional model. Although the method obtained reasonable results on experiments, there are some rooms to make improvements. Firstly, the results of labelling phase in their work totally based on matching functions of common vocabularies between text segments in an input text and attribute values of labels in knowledge base. In other words, it exploits the content-related feature of the text segments. This could not be effective, especially for some simple data types such as date-time, phone numbers. Instead, format-related features of data could be exploited to improve the performance of matching phase. Secondly, to revise the labels of text segments in reinforcement phase, the positional model in previous work exploited the occurrences of labels in a particular position in different strings in an input list. In other words, a label of a text segment is revised via a fixed position of the label in different strings in the input list. We argue that since the labels of text segments to be revised in reinforcement step are generated from matching step through the usage of some matching functions. Therefore, ones cannot ensure that a majority of correct matched labels always occur in a fixed position in different strings in the input list. Instead, the rigid constraint on fixed positions of labels should be relaxed and the distribution of labels in the input text should be considered to determine the probability of occurrence of a label in a particular position.

Those disadvantages motivated our work, which is an improvement of the study of Cortez et al. (2010). On the one hand, we exploit both format-related features and content-related features and incorporate them in a type-based similarity model for matching values in matching phase. On the other hand, we propose a novel proximity-based positional model for labels which exploits the proximity of labels in different positions in an input list. In next sections, we present our proposed techniques in detail.

3 Proximity-based positional model for labels

Cortez et al. (2010) proposed a positional model by considering the number of occurrences of a label in a fixed position in the input text. We argue that the information of a label in a fixed position in different lines of an input text may not be enough to determine a label of a text segment in the position. Since ones do not know in advance how many text blocks there are in each line of an input list and the text segments are labeled by matching functions, ones cannot ensure that a correct label always occurs in a fixed position in all lines.

As a simple example, ones can consider the labels *volume* in citation strings in bibliographic domain. Even they are written in the same orders of field values. The number of authors in those strings can be different, the lengths of paper titles and book titles can be different. Therefore, the positions of a correct label in different lines of an input list can be different. Ones can see another example in Figure 1. The figure illustrates an example of the results when we apply a labelling phase to have labels for text segments in an input list. In the example, when we consider the tex segments at the position two in a list, the number of labels "Author" are equal to the number of label "Title". Therefore, if we just consider the number of occurrences in a fixed position, we cannot determine the label of the text segment at the position two. Meanwhile, if we consider the occurrences of the labels "Author" in all positions in all lines of the list, we can see that it occurs frequently in positions around the position 2. That evidence of occurrences of labels "Author" near the position two should be taken into the probability to have the label "Author" in the position two. In other words, the neighbour positions of a label should be considered when we compute the probability of the label at a particular position. Based

on this idea, we propose a novel *proximity-based positional model* (PPM) for labels in which we exploit information of different positions of a considering label in the input text to compute the probability to have the label in a particular position.

3.1 Model formulation

The PPM for a label at a position would be estimated based on the propagated label counts from the labels at all other positions in an input text. Specifically, each label at each position of the input text is determined by the evidence of its occurrence to all other positions in the input text and the positions close to the label will get more share of the evidence than those far away. By this way, each position will receive propagated counts of labels from all positions in the input text. We formally define a *proximity-based positional model* (PPM) for labels as the following.

Given a list L including n lines $L = (l_1, l_2, ...l_n)$, let $T_L = (t_1, ..., t_i, ..., t_j, ..., t_N)$ is a list of all possible labels in the list L, N is obviously the number of labels in L.

$c(t, i)$: the number of times the label t occurs at position i in different lines of the list L.

$k(i, j)$: the discounting factor to the position i from a label at the position j. This factor can be any non-increasing function of $|i - j|$ and called a proximity-based density function. This means that $k(i, j)$ favours positions close to i. Several proximity-based density functions can be chosen to define $k(i, j)$. Each density function will lead to a specific PPM. We will analyse and explore different density functions in section 3.2.

$c'(t, i)$: the total propagated count of a label t at position i from the occurrences of the label t in all positions in the list L. We notice that even if $c(t, i)$ is 0, $c'(t, i)$ may be greater than 0. In other words, $c'(t, i)$ not only considers the positions of the label t at the position i, but also takes into account the neighbour positions of the label t via a proximity-based density function. Formally, $c'(t, i)$ is represented as in the equation 1.

$$c'(t, i) = \sum_{j=1}^{N} c(t, j)k(i, j) \tag{1}$$

From the label propagation function $c'(t, i)$, we have a label frequency vector $<c'(t_1, i), ..., c'(t_N, i)>$ at a position i. Accordingly, positional information of each label can be translated to label frequency information in this vector. Based on this formulation, we estimate a proximity-based positional model (PPM) of a label t at position i in a list L as in the equation 2.

$$p(t|L, i) = \frac{c'(t, i)}{\sum_{t' \in T_L} c'(t', i)} \tag{2}$$

where T_L is a set of labels in L and $c'(t, i)$ is defined by the equation 1.

According to the property of proximity-based propagation function, the value of $p(t|L, i)$ is mainly influenced by labels around the position i, not a fixed position in the list L. In other words, our model can measure positional information of a label and incorporate it into a language model. Moreover, if the value of σ is set to a small value, we would emphasise on local proximity of labels. The balance of local proximity evidence of labels in a string can be tuned by the parameter σ. Thus, our model can capture the proximity information of labels in a language modelling framework. Once we obtain a PPM for each position of labels, we use each PPM as a regular document language model for matching with a label of a text segment.

3.2 Proximity-based propagation functions for PPM

A major challenge in PPMs for labels is how to define the density function $k(i, j)$. Different kernel functions will lead to different PPM for labels. Following previous studies about computing the distances of words in a document in information retrieval, we consider five propagation functions $k(i, j)$ in our work and adapt the idea to labels in a document. Those functions are Gaussian kenel (eq. 3), Triangle kernel (eq. 4), Cosine kernel (eq. 5), Circle kernel (eq. 6), and Rectangle kernel (eq. 7) (De Kretser & Moffat 1999, Petkova & Croft 2007, Kaszkiel & Zobel 2001).

Examples of the curves of those kernel functions can be illustrated in the Figure 2. It can be seen in the figure that all the kernel functions have the range values from zero to one and they obtain the highest value when i equals to j. In the kernel functions, σ is a tuning parameter, which controls the spread of kernel curves to restrict the propagation scope of each label in a document. The optimal value of σ may vary according to different labels. If a label have wider semantic scope in a document, the value of σ should be larger. Due to proximity-based density function, PPM for labels allows us to explore the scope of positions of labels in a list L.

- Gaussian kernel:

$$k(i, j) = exp[\frac{-(i - j)^2}{2\sigma^2}] \tag{3}$$

- Triangle kernel:

$$k(i, j) = \begin{cases} 1 - \frac{|i-j|}{\sigma} & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- Cosine kernel:

$$k(i, j) = \begin{cases} \frac{1}{2}[1 + cos(\frac{|i-j|.\pi}{\sigma})] & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

- Circle kernel:

$$k(i, j) = \begin{cases} \sqrt{1 - (\frac{|i-j|}{\sigma})^2} & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- Rectangle kernel:

$$k(i, j) = \begin{cases} 1 & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Moreover, we can show that the positional model proposed in the study of Cortez et al. (2010) is actually a special case of our PPM when we set the value of σ to zero. In fact, when the value of σ equals to zero, the expression $|i - j| < \sigma$ returns true if $i = j$. In this case, proximity-based density function $k(i, j)$ is represented the equation 8.

$$k(i, j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Positions of segments	1	2	3	4	5	6	7	8	9
Line 1	**Author**	Title	**Author**	Title	Title	Booktitle	BookTitle	Year	Page
Line 2	?	**Author**	?	**Author**	Title	?	BookTitle	?	Page
Line 3	?	Title	**Author**	Title	Title	?	BookTitle	Page	
Line 4	**Author**	**Author**	?	Tittle	BookTitle	?	?	?	Year
Line 5	?	?	Title	Title	BookTitle	BookTitle	BookTitle	Year	?

Author?
Title?

Figure 1: An example of labels to be revised after matching phase

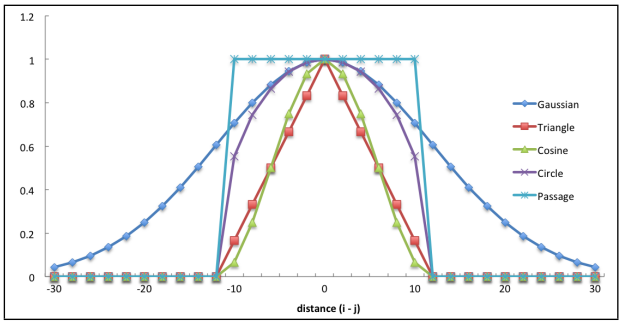

Figure 2: Proximity-based kernel functions ($\sigma = 12.0$)

From the equation 1, the proximity-based propagation function $c'(t,i) = c(t,i)$ for all the values of i. In other words, the value of proximity-based propagation function is equal to the number of times the label t occurs at position i in different lines of the list L. Therefore, we can conclude that the fixed-positional model in the study of Cortez et al. (2010) is a special case of our proximity-based positional model when we adjust the value of σ to zero.

4 Exploiting proximity-based positional model in IETS

In this section, we begin by stating the problem of IETS and then present the usage of proximity-based positional model and other techniques to improve the quality of extracting information.

4.1 Terminology and problem definition

Consider a list L of n lines, where each i^{th} line l_i is a string which represents an implicit unstructured record. For example, in bibliographic domain, each line l_i is a *reference* as a string to represent the field values of an academic publication. A typical representation of a publication may include information of authors, title, book title or publication venue, pages, date, volume, and some other information of a publication.

Formally, each a line l_i can be represented as $l_i = \{v_1 d_1 v_2 d_2 v_3 d_3 ...\}$, where v_i are *field values*, and d_i are *delimiter* or symbols to separate field values v_i. A *delimiter* is any character other than A..Z, a..z, or 0..9. The delimiters split a line into a sequence of *tokens*. The delimiters separate field values but may occur in field values. Each line is written in a particular *style* which defines the order of field values and delimiters to separate field values.

To assign labels for input list, we exploit a knowledge base which contains a set of pairs $K = \{<t_1, E_1>, ..., <t_m, E_m>\}$ where t_i is a distinct field or a label, E_i is a set of field values or occurrences of the field t_i. Given a list L of n lines including field values on a web page and a knowledge base KB, our goal for the problem of IETS is to extract automatically the field values in L and store them in a structured form, e.g a table, or an xml file.

4.2 Algorithm overview

Our method can be described in a sequence of operations over the input text. In general, the operations in our algorithm can be grouped into three main phases: splitting phase, matching phase, and refinement phase. In the splitting phase, each string in the input list is split into multiple text segments. In matching phase, we exploit a knowledge base to assign labels for text segments in the input list. In this phase, we devise a type-based similarity measure to evaluate how likely a text segment should have a label t in knowledge base. After matching phase, some text segments in the input list are unmatched with the knowledge base and therefore they do not have labels. Meanwhile, some other ones may have mismatched labels. These unmatched and mismatched labels will be revised by refinement phase. Due to this phase, they are detected and fixed if they are likely to be incorrect. To do that, proximity-based positional model proposed in section 3 is used to combined with sequential model to revise the results matching phase. Details about splitting, matching and refinement phase are accordingly described in section 4.3, 4.4, and 4.5.

4.3 Text-splitting phase

In this section we firstly present the text-splitting phase in our method to split input list into text segments, then explain the type-based similarity measure which we propose to label text segments. The algorithm of splitting text for each string in an input list is described in Algorithm 1. It is improved from the algorithm of block texts proposed in the study of Cortez et al. (2010) in which we consider format-related features to define regular expressions to segment and recognise the field values of some simple datatypes. In our work, we recognise eight primitive datatypes including numbers, date-times, page numbers, volumes and issues, URLs, email addresses, and phone numbers, then define regular expressions to segment texts in an input list. Due to this, we can obtain high performance on those simple datatypes and helps achieve effectiveness when we revise the labels of other field values in the input list in refinement phase. In the algorithm of splitting a line into text segments, we initially extract tokens from a string l based on the occurrence of white spaces. For each

token $t^{'}$ in a string l, we find a sequence of consecutive tokens starting from $t^{'}$ which satisfies any pre-defined regular expression to group them into a text segment. Moreover, if any two consecutive tokens co-occur in the same instance value according to a knowledge base, they will be in the same text segment. We notice that tokens which do not occur in the knowledge base are always in a single segment.

Input: A string l in an input list L, a knowledge base K, a set of regular expressions R for format-related features
Output: A set B of text segments
$T^{'} : <t_0^{'}, ..., t_n^{'}>$ = Extract all tokens from string l;
j = 0, i = 0;
while $j < n$ **do**
  k = j + 1;
  while $k < n$ **do**
    if $t_j^{'}...t_k^{'}$ *satisfies a regular expression in* R **then**
      $B_i = \{t_j^{'}...t_k^{'}\}$;
      i = i + 1;
      j = k +1;
    end
    k = k + 1;
  end
  $B_i = \{\} \cup <t_j^{'}>$;
  $C = \{<t, E> \in K, e \in E | t_j^{'}, t_{j+1}^{'} \in e\}$;
  if C *is not empty* **then**
    $t_j^{'}$ and $t_{j+1}^{'}$ co-occur ;
    $B_i = B_i \cup <t_{j+1}^{'}>$;
    j = j + 1;
  end
  i = i + 1;
  j = j + 1;
end

Algorithm 1: Algorithm for text-splitting phase

4.4 Matching phase

Given a text segment s, the purpose of labelling phase is to exploit a knowledge base to assign to s a label t based on the features of the string s. Therefore, we consider the process of of matching a string s and a label t as the process of checking whether the element s is a member of a set with the label t or not.

According to set theory, a membership relation between an element and a set can be described in two ways: intensional and extensional definition. Intensional definition of a set formulates its meaning by specifying all properties which an element must satisfy to reach the definition. For example, a year often has four digits and starts with the prefix "19" or "20". Similarly, the page numbers of a paper often starts with one of strings in the set {"page", "pages", "p", "pp", "pg"} and two numbers separated by a punctuation such as "-" or "~". The properties to define a set in intensional definition can be implemented in rules to verify whether an element is a member of the set or not. In our work, we formulate this type of features in a function f_r illustrated in the equation 9.

$$f_r(s,t) = \begin{cases} 1 & \text{if } \exists r \in R_t : r(s) = true \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where R_t is a set of rules for the label t to check whether the string s has a membership relation with t or not. The function *r(s)* is the application of the rule r on the string s. If there is a rule r in R_t can be applied to the string s, the function $f_r(s,t)$ returns one, otherwise, it returns zero.

Meanwhile, extensional definition describes a set by specifying every element in the set. In this way, we consider each label t as a set and its field values are elements of the set. Therefore, the similarity between a string s and a label t can be computed via the instance values of the label t in a knowledge base. Based on the values of the labels, we exploit the common tokens/q-grams shared between a string s and the values of a label t to define the similarity function between them. There are several ways to define this function. One of way is to view each text segment s as a query and all tokens in the instance values of the label t as a document, then we can apply any ranking function in information retrieval to measure and rank a label t by the relatedness between the query s and the label t. Similar to (Cortez et al. 2010), we define the matching function for this content-related feature as in the equation 10.

$$f_c(s,t) = \frac{\sum_{w \in s} fitness(w,t)}{|s|} \quad (10)$$

The fitness scores are computed for all tokens/q-grams w in the query string s, and the label t and it is defined as in the equation 11.

$$fitness(w,t) = \frac{freq(w,t)}{freq(w)} \times \frac{freq(w,t)}{freq_{max}(t)} \quad (11)$$

where $freq(w,t)$ is the number of values of the label t containing the token w, $freq(w)$ is the total number of instance values in the knowledge base containing the token w, and $freq_{max}(t)$ is the highest frequency of any token in instance values of the label t. The first fraction in the equation 11 represents the probability to have the token w in the type t. Given a token w, the first fraction returns the same value if the frequencies of the token in instance values of two different labels are alike. Therefore, the second fraction is used as a normalisation factor to take into account the importance of a token for a label. A token will be more important for a label if it occurs in several instance values of that label as compared to other tokens.

4.5 Refinement phase

The main purpose of refinement phase is to revise the results of the labelling phase to give labels for unmatched segments and rectify mismatched ones. Cortez et al. (2010) exploited the transitions of labels in an input text to revise the labels. This strategy is based on an assumption that the number of correct labels are more than incorrect ones in an input list. Moreover, it assumes that incorrect labels do not occurs within the same record. Therefore, statistical analysis on labels enables us to detect incorrect ones and fix them. A graphical model is built to represent the likelihood of transitions labels in the input text. For example, if there are several transitions from the label "author" to the label "year", the probability to revise an unknown label before the label "year" should be higher than probability to have other labels.

A sequential model (SM) for labels in an input list is defined as the following:

- A set of states $T = \{begin, t_1, t_2, ..., t_N, end\}$ where each state t_i describes an entity type labeled to a substring.
- A matrix A where the element a_{ij} is the probability of making a transition from state i to state j. Each element a_{ij} in the matrix A is defined as the equation 12.

$$a_{ij} = \frac{Number\ of\ transitions\ from\ state\ t_i\ to\ state\ t_j}{Total\ number\ of\ transitions\ out\ of\ state\ t_i} \quad (12)$$

An example of a sequential model in bibliographic domain can be seen in Figure 3. A sequential model is used to revise labels in the results of matching step and helps to improve the recall of extraction results. However, since the order of field values can be different in each line of input list and each line may contain several text segments, the usage of sequential model can decrease the precision of the system. For example, in bibliographic domain, a year can occur both after author names and conference names of different strings on a web page. Therefore, when we use only sequential model to revise an unknown text segment before a year, it may be labelled it as an "author" instead of "conference" if the transition from "author" to "year" is more popular. To deal with this problem, we exploit our proposed proximity-based positional model to determine the labels of text segments.

The proximity-based positional model for labels can be defined as a matrix P where the entry p_{jk} denotes the probability of the label t_j appearing at the *k-th* position in a line of input list. Formally, the value of p_{jk} is defined as in the equation 13.

$$p_{jk} = p(t_j|L, k) \quad (13)$$

To compute the probability to have a label t for a text segment, we combine matching score, sequential model score and proximity-based positional model score by using Bayesian disjunctive operator as in equation 14.

$$sim(s,t) = 1 - (1 - f_r(s,t)) \times (1 - f_c(s,t)) \times (1 - f_s(s,t)) \times (1 - f_p(s,t)) \quad (14)$$

$$f_s(s,t) = a_{ij} \quad (15)$$

$$f_p(s,t) = p_{jk} \quad (16)$$

The value of $f_s(s,t)$ and $f_p(s,t)$ are accordingly defined by equation 15 and 16. In the equation 15, i is the index of the label t in a list of labels T, j is the index of the label of the next segment of s. In the equation 16, j is the index of t in T and k is the position of s in an input string. The value of a_{ij} and p_{jk} are accordingly defined by sequential model as in the equation 12 and proximity-based positional model in the equation 13. Both matrixes A and P in both models are built directly by a single pass on the input list.

5 Experiments and results

In this section, we present our experiments to evaluate our method on real datasets to show that our proposed method can achieve better performance than the current state-of-art method of Cortez et al. (2010). We firstly describe the experimental setup and metrics for evaluations. Then we report experimental results and compare with previous work.

5.1 Data settings

We run experiments on the public datasets in two domains: *bibliographic* and *addresses* domain. In each domain, we build a knowledge base and testing data from different data sources. Firstly, in bibliographic domain, we use datasets from experiments in previous studies. They are *CORA* collection (Peng & McCallum 2006), and PersonalBib dataset (Mansuri & Sarawagi 2006, Zhao et al. 2008). We note that the experimental results our paper are performed on whole 500 citations in Cora dataset, not only 150 citations as reported in the paper of Cortez et al. (2010). In the domain *Addresses*, we download two datasets *BigBook* and *LARestaurants* from the RISE repository (RISE 1998) and then manually label the field values in each dataset. Detailed information about these datasets in both domains is summarised in table 1.

5.2 Metrics for evaluation

In the experiments, we verify the extraction results in each phase and evaluate how much our proposed techniques can give better performance than techniques used in the study of Cortez et al. (2010), the current state-of-art study on IETS. In the evaluation, we utilise well-known precision, recall, and F1 measure to compare.

We denote A_i as a referent set and B_i as testing results to be compared with A_i. The precision (P_i), recall (R_i) and F1 measure (F_i) are accordingly defined as in the equation 17, 18, and 19. In our experiments, A_i is the set of tokens which compose values with a label and B_i is a set of terms assigned to a corresponding label by our method.

$$P_i = \frac{|A_i \cap B_i|}{|B_i|} \quad (17)$$

$$R_i = \frac{|A_i \cap B_i|}{|A_i|} \quad (18)$$

$$F_i = \frac{2 \times P_i \times R_i}{P_i + R_i} \quad (19)$$

5.3 Experimental results and evaluation

In this section, we present the experimental results on both domains bibliographic and addresses and compare our results with the study of Cortez et al. (2010).

5.3.1 *Bibliographic domain*

Table 2 shows experimental results when we segment citation strings by matching phase only (M), matching and sequential model (M+SM), matching and reenforcement by fixed-positional model (M+PM), matching phase and reenforcement by fixed-positional and sequential model (M+SM+PM). We notice that we incorporates format-related features through rules in our matching phase. Therefore we can obtain high performance of extraction on simple data types such as page numbers, years, volumes and issues, as compared to what reported in the study of Cortez et al. (2010). We observe that the extraction process on those data types can be achieved above 94% of F1-measure. Therefore, in the next experiments, we consider how proximity-based positional model can improve the performance on three main labels including "Author", "Title", and "Booktitle".

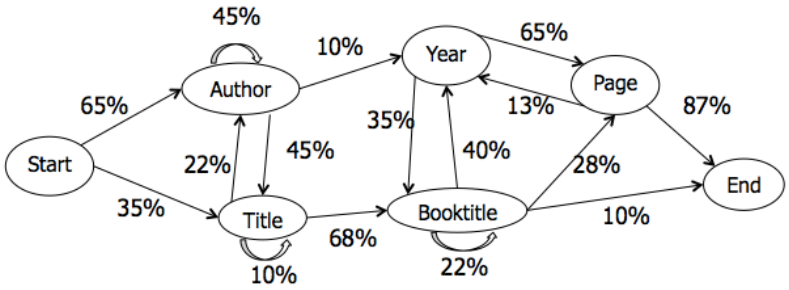

Figure 3: An example of a sequential model (SM)

Since the kernel function used to estimate the model can determine the performance of each strategy, we test our five proposed proximity-based kernel functions mentioned in 3.2 and conduct experiments to choose the best kernel function for our PPM for labels. To compare different kernel functions, we systematically vary the values of σ from 0 to 40 in the increments of 0.5 and observe the changes of the average F1 measure on the extraction of three labels "Author", "Title", and "Booktitle". The results of these experiments are illustrated in Figure 4.

Among all the kernel functions, we see that PPM with Gausian kernel gives the best performance as compared to other ones. Moreover, the peak value from the model can be obtained when the value of σ equals to three. The fact that Gausian kernel gives best performance can be explained is that the function has a special property: the propagated count drops slowly when the distance value $|i - j|$ is small, but drops quickly as the this distance value is large. This property is reasonable since the dependent labels in an input text often occur in an area near a particular position.

Moreover, to compare with the baseline and see how PPM can effectively capture proximity of labels in input text, we run experiments by using PPM with Gausian kernel ($\sigma = 3$) and compare with the baseline. The 4^{th} column (M+SM+PPM) in the table 3 illustrates the results of our method as compared to the baseline (M+SM+PM) when we set value of σ to three. In general, proximity-based positional model gives better final performance of labelling text segments as compared to fixed-positional model in previous work on bibliographic domain.

5.3.2 *Addresses domain*

Similar to bibliographic domain, we repeat the experiments in our method and compare with pervious work on Addresses domain. The experiments show that we can obtain 98.11% of F1-Measure by exploiting format-related features which are implemented in some simple regular expressions to recognise phone numbers. We also vary the values of σ and see how PPM model can improve performance of extraction as compared to the study of Cortez et al. (2010). It is interesting that the best final performance of our method when we use PPM model with different kernels is similar to performance we obtain by using a fixed positional model. That best performance is obtained when we set σ to be zero. As we have proven in the section 3.2, when fixed-positional model is actually a case of our model when we set the value of σ to be zero. Therefore, the results in two model in that case are similar.

The results of the performance can be explained by following reasons. Firstly, different from *bibliographic domain*, we notice that the dataset *LARestaurants* in *Addresses* domain is quite regular. Each field value includes only few tokens and the lengths of lines and field values in the dataset are quite similar. All address strings are written in a single order of field values.

Moreover, after using a simple application to count the common tokens in different field values within *Bigbook* dataset and *LARestaurants* dataset, we see that there is no overlapping token between the values of any two different fields in both knowledge base and testing data. In other words, the tokens in different fields are totally distinct. Therefore, when we perform matching phase, all text segments are assigned to a correct label or an empty label. In addition, the number of tokens in the values of each field in testing data are similar. Therefore, the positions of text segments given the same labels are similar. Meanwhile, in bibliographic domain, citation strings can have different authors, different of length of paper title. Therefore, the positions of text segments of a label can be different.

6 Conclusion

In this paper, we propose two novel techniques to improve the quality of information extraction by text segmentation. On the one hand, instead of exploiting matching based functions on values as in the study of Cortez et al. (2010), we propose a novel similarity measure in which we incorporate both format-related and content-related features into a type-based similarity measure. By combining both features into a single similarity measure, we achieve better performance on some primitive datatypes. This helps to improve the performance of revising steps. On the other hand, instead of considering the fixed positions of labels in reinforcement step as in previous work, we relax that constrains by our proximity-based approach. We consider related positions of labels appearing in different positions in an input list and combine it with sequential model to improve the quality of extraction model. In our work, we propose and study five different proximity-based density functions to estimate the proximity-based positional model. Experimental results show that the Gaussian density kernel achieves best performance and the results of our method yield higher performance than the state-of-art method.

The proposed technique opens some interesting future research directions. One of interesting directions is to study how to tune the value of σ automatically based on the statistics of labels in an input list. That is one of the future studies which we are investigating.

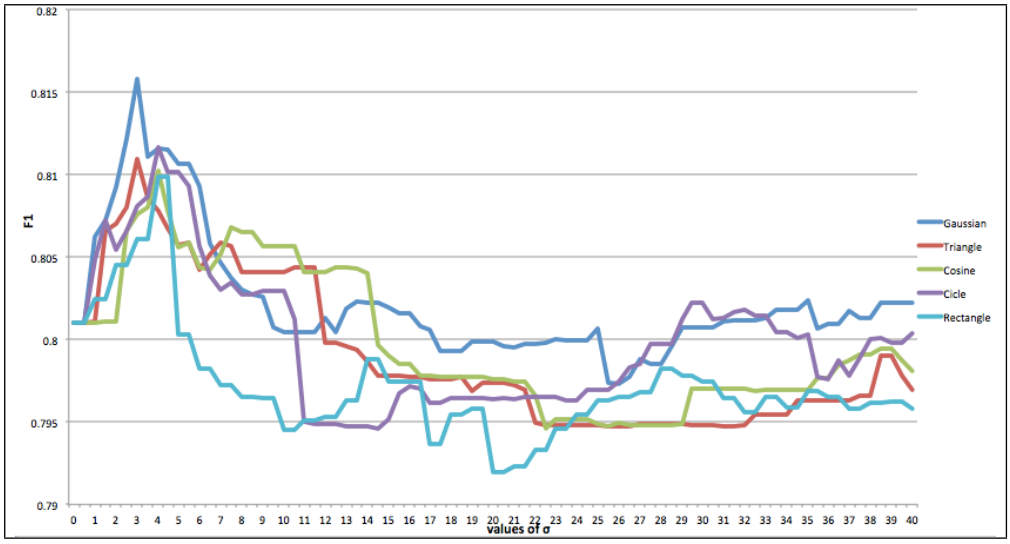

Figure 4: Performance on Cora dataset with different propagation kernel functions

7 Acknowledgement

We would like to thank Eli Cortez for explaining some technical details of experiments in the paper of Cortez et al. (2010).

References

Agichtein, E. & Ganti, V. (2004), Mining reference tables for automatic text segmentation, *in* 'Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 20–29.

Arasu, A. & Garcia-Molina, H. (2003), Extracting structured data from web pages, *in* 'Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data', pp. 337–348.

Borkar, V., Deshmukh, K. & Sarawagi, S. (2001), Automatic segmentation of text into structured records, *in* 'Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data', pp. 175–186.

Cortez, E., da Silva, A. S., Gonçalves, M. A. & de Moura, E. S. (2010), Ondux: on-demand unsupervised learning for information extraction, *in* 'Proceedings of the 2010 international conference on Management of data', SIGMOD '10, ACM, New York, NY, USA, pp. 807–818.

Crescenzi, V., Mecca, G. & Merialdo, P. (2001), Roadrunner: Towards automatic data extraction from large web sites, *in* 'Proceedings of the 27th International Conference on Very Large Data bases', pp. 109–118.

De Kretser, O. & Moffat, A. (1999), Effective document presentation with a locality-based similarity heuristic, *in* 'Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval', SIGIR '99, ACM, New York, NY, USA, pp. 113–120.

Freitag, D. & McCallum, A. (2000), Information extraction with hmm structures learned by stochastic optimization, *in* 'Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence', AAAI Press, pp. 584–589.

Guo, J., Xu, G., Cheng, X. & Li, H. (2009), Named entity recognition in query, *in* 'SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval', ACM, New York, NY, USA, pp. 267–274.

Kaszkiel, M. & Zobel, J. (2001), 'Effective ranking with arbitrary passages', *Journal of the American Society for Information Science and Technology* **52**(4), 344–364.

Lafferty, J. D., McCallum, A. & Pereira, F. C. N. (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *in* 'Proceedings of the 18th International Conference on Machine Learning', pp. 282–289.

Mansuri, I. R. & Sarawagi, S. (2006), Integrating unstructured data into relational databases, *in* 'Proceedings of the 22nd International Conference on Data Engineering', pp. 29–40.

Peng, F. & McCallum, A. (2006), 'Information extraction from research papers using conditional random fields', *Information Processing and Management* **42**, 963–979.

Petkova, D. & Croft, W. B. (2007), Proximity-based document representation for named entity retrieval, *in* 'Proceedings of the sixteenth ACM conference on Conference on information and knowledge management', CIKM '07, ACM, New York, NY, USA, pp. 731–740.

RISE (1998), Rise - a repository of online information sources used in information extraction tasks, http://www.isi.edu/info-agents/rise/index.html.

Sarawagi, S. (2008), 'Information extraction', *Foundation and Trends in Databases* **1**(3), 261–377.

Seymore, K., Mccallum, A. & Rosenfeld, R. (1999), Learning hidden markov model structure for information extraction, *in* 'AAAI99 Workshop on Machine Learning for Information Extraction', pp. 37–42.

Zhao, C., Mahmud, J. & Ramakrishnan, I. V. (2008), Exploiting structured reference data for unsupervised text segmentation with conditional random fields, *in* 'Proceedings of the SIAM International Conference on Data Mining', pp. 420–431.

Domain	Dataset	Attributes	Records
Bibliographic Data	*Cora*	13	500
	PersonalBib	7	395
Address Data	*Bigbook*	5	4,000
	LARestaurants	4	250

Table 1: Domains and datasets used in our experiments.

Field	Matching(M)	M+SM	M+PM	M+SM+PM
Author	0.7080	0.7548	0.6774	0.7845
Title	0.7882	0.7650	0.6331	0.8217
Booktitle	0.7971	0.8609	0.6375	0.7967
Pages	0.9961	0.9961	0.9961	0.9961
Year	0.9912	0.9912	0.9912	0.9912
Volume	0.8483	0.9787	0.7880	0.9404
Issue	0.9663	0.9663	0.9663	0.9663

Table 2: Experimental results on *Cora* dataset.

Field	M	M+SM+PM	M+SM+PPM
Author	0.7080	0.7845	0.7949
Title	0.7882	0.8217	0.8457
Booktitle	0.7971	0.7967	0.8068
Pages	0.9961	0.9961	0.9961
Year	0.9912	0.9912	0.9912
Volume	0.8483	0.9404	0.9818
Issue	0.9663	0.9663	0.9663

Table 3: Experimental results on *Cora* dataset.

Field	M	M+SM	M+PM	M+SM+PM	M+SM+PPM
Name	0.6182	0.8148	0.8903	0.9724	0.9724
Street	0.9073	0.8298	0.8591	0.9808	0.9808
City	0.7388	0.9845	0.8388	0.9857	0.9857
Phone	0.9811	0.9874	0.9817	0.9923	0.9923

Table 4: Experimental results on *LARestaurants* dataset.