*Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 1910–1917
Marseille, 11–16 May 2020


# Headword-Oriented Entity Linking: A Special Entity Linking Task with Dataset and Baseline

**Mu Yang[†], Chi-Yen Chen[†], Yi-Hui Lee[†], Qian-Hui Zeng[†], Wei-Yun Ma[*†], Chen-Yang Shih[‡], Wei-Jhih Chen[‡]**

[†]Institute of Information Science, Academia Sinica, Taipei, Taiwan
[‡]R&D Center, PIXNET Corporation, Taipei, Taiwan
emfomy@gmail.com, chiyench@usc.edu, yi-hui.lee@utdallas.edu,
littlemiss330@gmail.com, ma@iis.sinica.edu.tw, kent@pixnet.tw, wayne@pixnet.tw

**Abstract**

In this paper, we design headword-oriented entity linking (HEL), a specialized entity linking problem in which only the headwords of the entities are to be linked to knowledge bases; mention scopes of the entities do not need to be identified in the problem setting. This special task is motivated by the fact that in many articles referring to specific products, the complete full product names are rarely written; instead, they are often abbreviated to shorter, irregular versions or even just to their headwords, which are usually their product types, such as "stick" or "mask" in a cosmetic context. To fully design the special task, we construct a labeled cosmetic corpus as a public benchmark for this problem, and propose a product embedding model to address the task, where each product corresponds to a dense representation to encode the different information on products and their context jointly. Besides, to increase training data, we propose a special transfer learning framework in which distant supervision with heuristic patterns is first utilized, followed by supervised learning using a small amount of manually labeled data. The experimental results show that our model provides a strong benchmark performance on the special task.

**Keywords:** Corpus, Information Extraction, Distant Supervision

## 1. Introduction

Named entity linking, or entity linking (EL), is determining the identity of entities that are mentioned in the text and bridging them from unstructured textual data with structural knowledge bases. Traditionally, entity linking involves two sequential tasks: first, detecting mentions using named entity recognition, seeking to locate entity chunks in text and classify their entity types; second, linking the recognized entity mention with the corresponding entry in a knowledge base.

Domain-specific entity linking has attracted attention due to commercial demands in practice, especially product name linking. Unlike conventional named entities such as person, location, or organization names, complete full product names are rarely written in context; instead, they are often abbreviated to a shorter, irregular version or even just to their headwords. For instance, in a real blog article about "Dior Addict Lacquer Stick" (PID 17755 in our cosmetic database) and other cosmetic products, the 'PID 17755' item is sometimes written as "Dior lacquer stick", or just "pen-shaped lacquer stick" or even abbreviated to its product type — "stick", such as "I really like the stick that Dior released last month," where "stick" refers to 'PID 17755'. In most cases, the mentions contain at least the product type to represent the product. Such being the case, from a practical application perspective such as product recommendation, the key goal is to link each product-type word, such as "stick", "mask", "eyeshadow", etc., to knowledge bases if they do represent certain products in the context without the need for named entity recognition.

Based on this concept, in this paper, we design headword-oriented entity linking (HEL), a specialized entity linking problem in which only the headwords of the entities are to be linked to knowledge bases; the mention scopes of the entities do not need to be identified in the problem setting. HEL can be seen as a distinctive integration of traditional EL and coreference resolution.


Figure 1: Headword-Oriented Entity linking — only link "唇釉" (Stick, the red text in the figure) to an unique knowledge base ID, instead of linking its whole named entity — "DIOR癮誘超模漆光唇釉" (Dior Addict Lacquer Stick, the green background in the figure) — to the ID.

To fully design this special task, we construct a labeled cosmetic corpus as a public benchmark[1] for this problem. We collect blog articles from PIXNET[2] and seek to link cosmetic products in the articles to the cosmetics database, as shown in Fig. 1. Most cosmetics are not written in their

[*]Corresponding author

[1]The package and dataset are available at https://github.com/ckiplab/cosmel.

[2]One of the largest online Taiwanese blog sites and social networking service companies.

full product names. Word deletions, replacements, and insertions are very common in these articles. Also, since not every product-type word represents a specific product, we use two other labels — one is **GP** (general product), representing an uncertain product, such as "I never use sticks". The other is **OSP** (other specific products), meaning it is indeed a particular product but is not listed in our database (no entry in the database).

In this paper, we also propose a product embedding model for this task. Each product corresponds to a dense representation that jointly encodes not only its context but also various information, including its full product name and official advertised description.

Additionally, to increase training data, a special transfer learning framework is applied — several heuristic patterns are first used to generate noisy labeled data for all articles; these are then used to distantly supervise the model (Mintz et al., 2009) as a pretraining phase. After that, fully supervised learning based on a small amount of manually labeled data is further applied to fine-tune the model.

Our contributions are five-fold:

- We design headword-oriented entity linking, a special named linking problem in which mention scopes are not required to be identified. This setting is especially useful for product linking.
- We create a labeled cosmetic corpus as a public benchmark for this problem.
- We propose a product embedding model to address the problem as a strong baseline, where diverse information about products is jointly encoded using dense representations.
- We present a special transfer learning framework, involving distant supervision of the model with a large number of noisy labels, and then supervised learning using a small number of manually labeled data.
- To the best of our knowledge, we are the first to study entity analysis in the cosmetic domain.

## 2. Related Work

The two main strategies for entity linking are discrete feature-based and embedding-based systems. Shen et al. (2015) systematically organize discrete feature-based entity linking systems. Discrete features include name string comparisons (Liu et al., 2013) and similarities of bags of words (Lin et al., 2012), concept vectors (Chen and Ji, 2011), and other manually designed features (McNamee et al., 2009). With these discrete features, supervised learning models such as binary classifiers (Zhang et al., 2010) and learning-to-rank methods (Kulkarni et al., 2009) have been implemented for entity ranking.

However, featured-based entity linking systems are highly data-dependent, which requires extensive effort to design domain-specific features. Moreover, as the discrete features are often too sparse to train the model, they are unlikely to apply to different domains. Recently, many approaches have devised to learn representations of the entity and use it for entity linking. Gupta et al. (2017) build an embedding-based linking system that learns representations for each entity without domain-specific training data or hand-engineered features. Yamada et al. (2016) learn word and entity embeddings for named entity disambiguation based on the skip-gram model. Francis-Landau et al. (2016) utilize convolutional neural networks to capture semantic correspondence between a mention's context and a proposed target entity. Sun et al. (2015) disambiguate entities using the mention embedding (the average of the embeddings of the words it contains), the context embedding by a convolutional neural network (CNN), and entity embedding through entity surface words and entity class from the knowledge base.

Conventionally, supervised models have been used for entity linking. However, one significant problem with supervised approaches is their heavy reliance on large amounts of annotated training data. Moreover, entity linking annotation is expensive and time-consuming. Some supervised approaches train their models on a small manually-created data set consisting of thousands of labeled entity mentions (Shen et al., 2012; Dredze et al., 2010; McNamee, 2010; Li et al., 2009). Some systems (Bunescu and Paşca, 2006; Agirre et al., 2009) use hyperlinks in Wikipedia articles to construct training data.

For all these entity linking researches, mention scopes need to be identified by named entity recognition, but the scope identification is not always indispensable in terms of practical needs for applications, such as product recommendation or product analysis. That motivates us to design this special task based on headword-oriented entity linking in this paper.

Our special task can be regarded as nominal coreference (Fonseca et al., 2018) and traditional entity linking. Such mentions with complex lexicon syntactic patterns are linked together as coreference chains before being linked to unique data or knowledge base IDs. One can also use that pipeline to tackle this problem, but in our design, we adopt a different strategy — we proposed a knowledge base driven approach in order to directly tackle the task without explicit coreference resolution.

## 3. Corpus Construction

We collected over $5{,}000$ products and about $50{,}000$ blog articles from the PIXstyleMe[3] web service. These articles contain a total of over 5 million sentences and over 41 million words. We standardized the text in the database and applied word segmentation. We also created repositories of products, brands, and product types as knowledge bases.

### 3.1. Product Database

**Product** We created a product database from PIXstyleMe's database consisting of product names, brands, descriptions, etc. After manually removing duplicate products and fixing typos, we collected $5{,}060$ products in the database, and assigned a unique ID (product ID, PID) for each product.

[3]A fashion community of PIXNET, `https://styleme.pixnet.net`

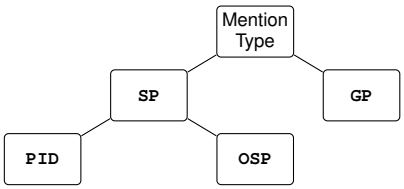



Figure 2: Mention types: **SP** (Specific Product), **PID** (Product with ID), **OSP** (Other Specific Product), and **GP** (General Product)

**Brand** We also collected the aliases (including English and Chinese names) of each brand (364 in total) from the database. For example, both "sk2" and "skii" refer to the brand "SK-II"; "shuuemura", "shu uemura", and "植村秀" represent the brand "Shu Uemura".

**Headword** We collected 926 headwords, each of which represents a type of product. Examples are "面膜" (facial mask), "唇膏" (lipstick), and "香水" (perfume).

**Word Segmentation** After the above preprocessing, we applied word segmentation using CKIPWS (Ma and Chen, 2003a; Ma and Chen, 2003b) on product names and their descriptions. In order to get better word segmentation, we add many cosmetic specific words into our lexicon.
We also make some necessary modifications to the ambiguous terms after word segmentation. For example, "修容蜜粉餅" can be segmented as "修容" (contouring) + "蜜粉餅" (pressed powder cake) or "修容蜜" (liquid blush) + "粉餅" (powder cake). Though the previous one is more semantically reasonable, the word segmentation tool is not able to determine it since all the above four words are collected in the lexicon. Therefore, we add "修容蜜粉餅" to the lexicon, and segment it using regular expressions after word segmentation. We added about a hundred such terms into our dataset. With the above trick, all the headwords in product names are segmented correctly. This is a crucial step for mention detection (§3.2.).

### 3.2. Mention Detection

In conventional entity linking, most entities have a specific scope and can be differentiated as people, time, or organizations. However, cosmetic entities are different from general named entities. To be more specific, most cosmetic entities are long noun phrases, such as "迪奧輕透光空氣蜜粉" (PID 6064, Diorskin, Nude Air Loose Powder). Also, most of the time, while being mentioned, entities do not appear as the full name — only 20,617 of a total 967,969 mentions are written as complete full product names. For instance, both "輕透光空氣蜜粉" (nude air loose powder) and "空氣蜜粉" (air loose powder) can be linked to the product "迪奧輕透光空氣蜜粉" (Diorskin Nude Air Loose Powder). So in our task, the headwords are used to represent the corresponding mentions, and the headwords are to be linked to knowledge bases.

**Mention Type** As shown in Fig. 2, we classify the mentions into several types: **PID**, **OSP**, and **GP**. **PID** (Product with ID) mentions are linked to specific products in the database. **OSP** (Other Specific Product) mentions are also specific products but are not contained in the database. **GP** (General Product) mentions are general concepts or plural forms. **OSP** and **GP** correspond to **NIL** mentions in conventional entity linking.

| Dataset | Total | PID | OSP | GP |
|---|---|---|---|---|
| *RLabel* | 906,585 | 94,826 | 195,307 | 616,452 |
| *GLabel* | 40,970 | 5,778 | 11,469 | 23,723 |

Table 1: Dataset mentions

### 3.3. Headword-Oriented Entity Linking

To reflect the different characteristics of cosmetic data, we redefine the scope of entities. Instead of detecting whole noun phrases, we simply detect the headword of the phrase, for instance, "蜜粉" (loose powder) for the above example. In this way, 967,969 mentions are detected in our corpus.
We called this method headword-oriented entity linking (HEL). With this special idea, the time spent in the annotation of the corpus will be much less than conventional entity linking. This argument would stand for any product types besides cosmetics domain. That means it will be much easier and faster to prepare a training dataset for a new domain. Without HEL, annotators need to read all sentences (over 5 million sentences) in the corpus, whereas our annotators only need to read those sentences containing headwords (only 800K sentences). Please note that our goal is to let the special task only focus on relatively easy but the most popular, critical cases, in order to prepare the training set for a new domain quickly. Aliases without headword are left to other approaches/tasks, such as coreference resolution, to identify.

### 3.4. Annotation and Heuristic Rule

Although there is no well-labeled data set in the cosmetics domain, it is not practical to manually label the entire corpus. To increase training data, a special transfer learning framework is presented. In essence, we first design several simple heuristic patterns (Heuristic Rule, a rule-based method) using regex and string matching based on observation to generate noisy labeled data for all articles, denoted as the *RLabel* corpus, and then use them to supervise the model in a pretraining phase distantly.
After that, to fine-tune the model, we further apply fully supervised learning based on a small amount of manually labeled data, denoted as the *GLabel* (golden label) corpus.
As an example of a heuristic pattern, noun phrases after "這款" (this; note that "款" is a quantifier in Chinese that widely used for cosmetic products) are likely to be cosmetic products. For example, the phrase "這款面膜" (this facial mask) usually refers to a facial mask product occurred previously, while "一款面膜" (a facial mask) is generally used as a general concept.
The distributions of the *RLabel* and *GLabel* corpora are given in Tab. 1.

## 4. Product Embedding Model

We seek to build a baseline model for the special task. The model aims to link each cosmetic product's headword to the corresponding product entry in the knowledge base. If

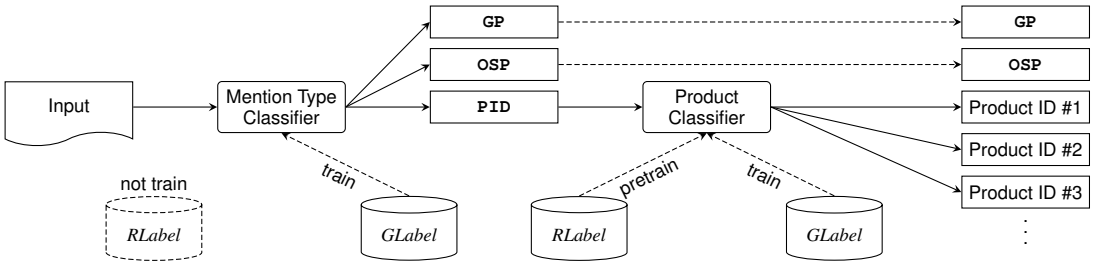



Figure 3: The pipeline — the mention type classifier (§4.1.) and the product classifier (§4.2.).

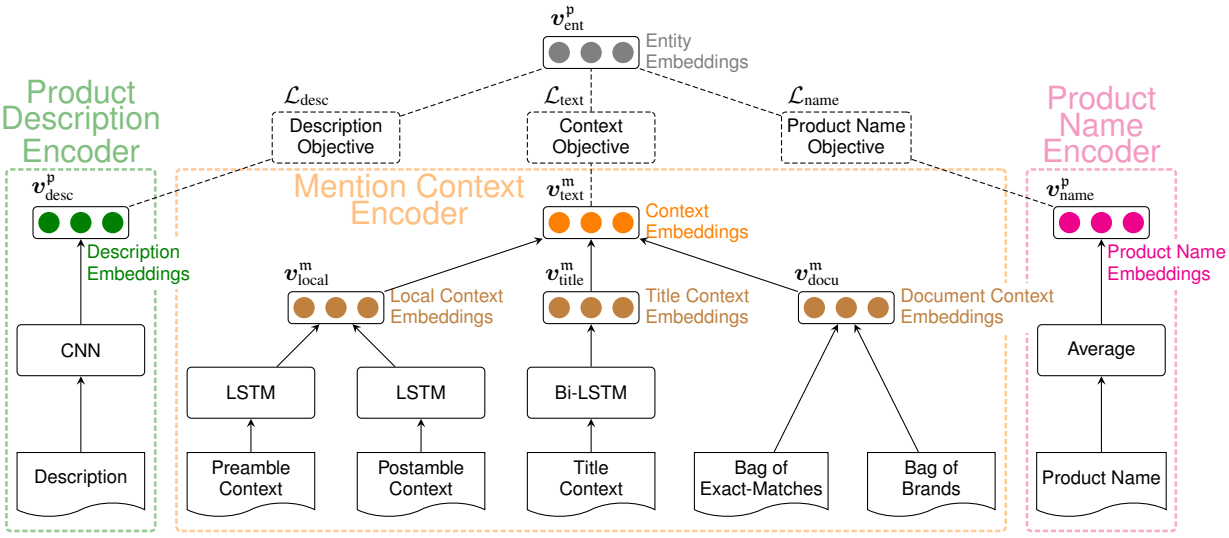



Figure 4: Product classifier (§4.2.) and its modules — the product description encoder (**D**, §4.3.2.), the mention context encoder (**C**, §4.3.1.), and the product name encoder (**N**, §4.3.3.).

it refers to a product not in our database, it is labeled **OSP**; if it is a general concept, it is marked **GP**.

To jointly consider various features of products, we follow the idea of joint encoding (Gupta et al., 2017) and present a product embedding model for our problem. Each product is represented as a dense representation to jointly encode products' context and various information about the products, including their advertised description and product names. We define $\mathcal{P}$ as the set of cosmetic products, $\mathcal{B}$ as the set of brands, $\mathcal{T} = \{\texttt{PID}, \texttt{OSP}, \texttt{GP}\}$ as the set of mention types, $\mathcal{M}$ as the set of mentions, and $\mathcal{V}$ as the set of vocabularies.

In our experiment, we found that a single model is not able to solve the problem, as the number of mentions labeled **OSP** and **GP** covers around 90% of the mentions, whereas **PID** contains 5,060 different products but only covers around 10% of the mentions; thus the model would be forced to pay more attention to the recall of **GP**/**OSP** even sacrificing the precision of other labels (the products with ID). With a single model only, it would gain a total accuracy of 77%, with 91% precision on **GP**, 78% on **OSP**, and 0% on all product IDs. However, product IDs should be more important for practical needs.

To address this problem, a pipeline strategy (Fig. 3) is introduced: we first use a mention type classifier (§4.1.) to classify the mentions into the three mention types, and then apply a product classifier (§4.2.) to link further those **PID** mentions to cosmetic products in our database.

## 4.1. Mention Type Classifier

With the mention type classifier, we seek to distinguish **PID** (products with predefined ID), **OSP** (products not in our database), and **GP** (general concept).

Here we use a context encoder (§4.3.1.) to obtain the mention context embeddings $\boldsymbol{v}^{\mathfrak{m}}_{\text{text}}$ of a given mention $\mathfrak{m} \in \mathcal{M}$, and linearly project the embeddings onto a 3-dimensional vector $\boldsymbol{v}^{\mathfrak{m}}_{\text{mtype}} = (u^{\mathfrak{m}}_{\texttt{PID}}, u^{\mathfrak{m}}_{\texttt{OSP}}, u^{\mathfrak{m}}_{\texttt{GP}})^\top \in \mathbb{R}^3$, and apply cross-entropy loss.

## 4.2. Product Classifier

As shown in Fig. 4, to jointly encode the information of products, we employ three encoders: a mention context encoder (§4.3.1.), a product description encoder (§4.3.2.), and a product name encoder (§4.3.3.). For more information, see §4.3.

Given a product $\mathfrak{p} \in \mathcal{P}$, and any $\mathfrak{m}$ linked to this product, our target is to acquire a representation of the entity $\boldsymbol{v}^{\mathfrak{p}}_{\text{ent}} \in \mathbb{R}^D$ that is similar to the mention context embeddings $\boldsymbol{v}^{\mathfrak{m}}_{\text{text}}$, the product description embeddings $\boldsymbol{v}^{\mathfrak{p}}_{\text{desc}}$, and the product name embeddings $\boldsymbol{v}^{\mathfrak{p}}_{\text{name}}$ obtained by the above three encoders.

Precisely, we maximize the probability of predicting the correct product $\mathfrak{p}_{\mathfrak{m}} \in \mathcal{P}$ from a given mention $\mathfrak{m}$ as

$$P_{\text{text}}(\mathfrak{p}_{\mathfrak{m}}|\mathfrak{m}) = \frac{\exp\left(\boldsymbol{v}^{\mathfrak{p}_{\mathfrak{m}}}_{\text{ent}} \cdot \boldsymbol{v}^{\mathfrak{m}}_{\text{text}}\right)}{\sum_{p \in \mathcal{P}} \exp\left(\boldsymbol{v}^{p}_{\text{ent}} \cdot \boldsymbol{v}^{\mathfrak{m}}_{\text{text}}\right)}, \tag{1}$$

and maximize the log-likelihood

$$\mathcal{L}_{\text{text}} = \frac{1}{|\mathcal{M}|} \sum_{\mathfrak{m} \in \mathcal{M}} \log P_{\text{text}}(\mathfrak{p}_{\mathfrak{m}} | \mathfrak{m}). \quad (2)$$

We use an objective similar to that above for product description and name embeddings; that is, we maximize the probabilities $P_{\text{desc}}(\mathfrak{p}|\boldsymbol{v}^{\mathfrak{p}}_{\text{desc}})$ and $P_{\text{name}}(\mathfrak{p}|\boldsymbol{v}^{\mathfrak{p}}_{\text{name}})$, defined similarly as eq. (1), and maximize the log-likelihoods $\mathcal{L}_{\text{desc}}$ and $\mathcal{L}_{\text{name}}$, defined similarly as eq. (2).
Finally, we maximize the summation of the objectives

$$\mathcal{L}_{\text{ent}} = \mathcal{L}_{\text{text}} + \mathcal{L}_{\text{desc}} + \mathcal{L}_{\text{name}} \quad (3)$$

to learn the product embeddings $\boldsymbol{v}^{\mathfrak{p}}_{\text{ent}}$ for each product $\mathfrak{p} \in \mathcal{P}$.

## 4.3. Encoder Modules

In this section, we describe several encoder modules used in our model. Following the idea of joint encoding in Gupta et al. (2017), we modify the mention context encoder (§4.3.1.) and the product description encoder (§4.3.2.). Furthermore, we propose an additional product name encoder (§4.3.3.) inspired by Sun et al. (2015).

### 4.3.1. Encoding the Mention Context — C

For mention context, we encode local, title, and document contexts, and combine them into mention embeddings.

**Local-Context Encoder** The sentence containing a mention is usually the most important for the corresponding cosmetic product. It might contain the name of the product and some description of it.
Given a mention word $\mathfrak{m} \in \mathcal{M}$, which denotes the sentence contains this mention as $s = (w_1, \ldots, \mathfrak{m}, \ldots, w_L)$, where $w_\bullet$ are the words and $L$ is the length of this sentence, we split the sentence into the preamble $\overrightarrow{s} = (w_1, w_2, \ldots, \mathfrak{m})$ and the postamble[4] $\overleftarrow{s} = (w_L, w_{L-1}, \ldots, \mathfrak{m})$. We apply two different long short term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) to both preamble and postamble contexts separately, using the pretrained word embeddings of each word as inputs. We concatenate the last hidden states $\overrightarrow{\boldsymbol{h}}_{\mathfrak{m}}, \overleftarrow{\boldsymbol{h}}_{\mathfrak{m}} \in \mathbb{R}^d$ of both LSTMs and pass them through a single-layer feed-forward network to produce the local context embeddings $\boldsymbol{v}^{\mathfrak{m}}_{\text{local}} \in \mathbb{R}^D$.

**Title-Context Encoder** The title of an article usually contains the article's topic. Most mentions in this article are related to this topic.
Similar to the local-context encoder, we use LSTM on the title of the article containing the given mention $\mathfrak{m}$. Here we use bidirectional LSTM, concatenate the last hidden states of both directions, and pass them through a single-layer feed-forward network to produce title context embeddings $\boldsymbol{v}^{\mathfrak{m}}_{\text{title}} \in \mathbb{R}^D$.

**Document-Context Encoder** We use document-wise information to obtain document context embeddings. The first is the *exact match*. An exact match is a scope of characters in the article which uses exactly the same characters as one of the cosmetic product. We assume that the mention $\mathfrak{m}$ is related to one or more products mentioned in the article. Since we have high confidence in the exact matches, such information is valuable to producing embeddings. We use a bag-of-products representation $\boldsymbol{v}^{\mathfrak{m}}_{\text{exact}} \in \{0,1\}^{|\mathcal{P}|}$, similar to Lazic et al. (2015), when collecting the exact-matched products that appeared in the article.
Also, we assume that the mention $\mathfrak{m}$ is related to one of the brands mentioned in the article. Similarly, we use a bag-of-brands representation $\boldsymbol{v}^{\mathfrak{m}}_{\text{brand}} \in \{0,1\}^{|\mathcal{B}|}$, collecting all the brands that appear in the article.
To produce the document context embeddings $\boldsymbol{v}^{\mathfrak{m}}_{\text{docu}} \in \mathbb{R}^D$, we concatenate $\boldsymbol{v}^{\mathfrak{m}}_{\text{exact}}$ and $\boldsymbol{v}^{\mathfrak{m}}_{\text{brand}}$ and pass them through a single-layer feed-forward network.

**Mention-Context Encoder** Finally, we combine the above local $\boldsymbol{v}^{\mathfrak{m}}_{\text{local}}$, title $\boldsymbol{v}^{\mathfrak{m}}_{\text{title}}$, and document $\boldsymbol{v}^{\mathfrak{m}}_{\text{docu}}$ context embeddings by concatenating and passing them into a single-layer feed-forward network. The output embeddings are denoted as $\boldsymbol{v}_{\text{text}} \in \mathbb{R}^D$, containing all the information on the given mention $\mathfrak{m}$ (Fig. 4, middle).

### 4.3.2. Encoding the Product Description — D

The textual description of a cosmetic product provides its ingredients, usage, effect, and features. This information helps us to produce embeddings with cosmetic product knowledge bases.

**Product Description Encoder** Given a product $\mathfrak{p} \in \mathcal{P}$, denote the description of this product as $d = (w_1, \ldots, w_L)$. Similar to Francis-Landau et al. (2016), we apply a convolution neural network (CNN) on the sentence $d$, followed by a maximum pooling layer, using the pretrained word embeddings of each word as inputs. The CNN outputs are passed into a single-layer feed-forward network to produce the product description embeddings $\boldsymbol{v}^{\mathfrak{p}}_{\text{desc}}$ (Fig. 4, left).

### 4.3.3. Encoding the Product Name — N

The name of a cosmetic product is one of the unique features in cosmetic product entity linking. Unlike most entity linking, the cosmetic product names are usually very long, and are thus useful for entity recognition.

**Product Name Encoder** Denote the name of a product $\mathfrak{p}$ as $n = (w_1, \ldots, w_L)$. We average the pretrained word embeddings of each word $w_\bullet$ to obtain the product name embeddings $\boldsymbol{v}^{\mathfrak{p}}_{\text{name}}$ (Fig. 4, right).

## 4.4. Transfer Learning Based on Distant Supervision

To increase the amount of training data, we present a special transfer learning framework: we first distantly supervise the model with a large number of noisy labels (*RLabel*), and then use supervised learning with a small number of manually labeled data (*GLabel*). *RLabel* contains more data but is less reliable, whereas *GLabel* is more reliable but the size is limited.
In the experiments, the complete process is denoted as *RLabel+GLabel*; *RLabel* refers to using only noisy labels, and *GLabel* refers to using only manually labeled data.
We expect the *RLabel+GLabel* model to be able first to learn a big picture framed by the simple patterns and then fine-tune the model to capture recognition details provided

[4] The postamble context is reversed so that the LSTM starts at the last word $w_L$ and ends at the mention $\mathfrak{m}$.

| Metric | | HEL | | | Baseline |
|---|---|---|---|---|---|
| | | Dataset | | | Rule |
| | | *RLabel* | *GLabel* | *RLabel+GLabel* | |
| **Overall** | *Accuracy* | 66.31±0.23 | **84.15**±0.46 | 80.79±0.54 | 65.17 |
| | *F1 Score* | 64.29±0.18 | **84.16**±0.38 | 80.88±0.47 | 63.89 |
| **PID** | *F1 Score* | 61.34±0.35 | **76.39**±0.83 | 76.29±0.58 | 62.39 |
| **OSP** | *F1 Score* | 37.54±0.30 | **77.29**±0.50 | 72.01±0.47 | 38.49 |
| **GP** | *F1 Score* | 77.35±0.20 | **89.14**±0.41 | 86.04±0.55 | 75.98 |

Table 2: Accuracy and F-measure (mean and standard deviation, %) of mention type classifier

| Metric | | Modules | HEL | | | Baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Dataset | | | Similarity Classifier | | | Rule |
| | | | *RLabel* | *GLabel* | *RLabel+GLabel* | *Emb* | *Bag* | *Emb+Bag* | |
| **PID** | *Accuracy* | **C** | 65.94±0.66 | 78.96±0.65 | 85.14±0.57 | 58.00 | 56.07 | 57.06 | 53.20 |
| | | **C+D** | 65.55±0.34 | 79.71±0.73 | 84.62±0.36 | | | | |
| | | **C+N** | 65.66±0.69 | 79.81±0.94 | **87.51**±0.52 | | | | |
| | | **C+D+N** | 65.38±0.64 | 79.41±0.92 | 86.43±0.65 | | | | |
| | *F1 Score* | **C** | 64.83±0.56 | 76.94±0.71 | 83.43±0.64 | 62.78 | 61.20 | 62.09 | 57.06 |
| | | **C+D** | 64.34±0.39 | 77.71±0.71 | 82.82±0.47 | | | | |
| | | **C+N** | 64.99±0.77 | 77.74±0.96 | **86.52**±0.66 | | | | |
| | | **C+D+N** | 64.55±0.64 | 77.30±1.05 | 85.47±0.65 | | | | |

Table 3: **PID** accuracy and F-measure (mean and standard deviation, %) of product classifier. Here we test only data labeled with a PID.

| Metric | | Modules | HEL | | | | Baseline |
|---|---|---|---|---|---|---|---|
| | | | Dataset for Mention Type Classifier | | | | |
| | | | *GLabel* | | *RLabel+GLabel* | | Rule |
| | | | Dataset for Entity Embeddings Model | | | | |
| | | | *GLabel* | *RLabel+GLabel* | *GLabel* | *RLabel+GLabel* | |
| **Overall** | *Accuracy* | **C** | 82.77±0.55 | 83.21±0.51 | 79.34±0.55 | 79.96±0.53 | 64.92 |
| | | **C+D** | 82.82±0.55 | 83.20±0.53 | 79.42±0.55 | 79.93±0.53 | |
| | | **C+N** | 82.81±0.59 | **83.40**±0.51 | 79.43±0.53 | 80.18±0.53 | |
| | | **C+D+N** | 82.81±0.54 | 83.34±0.52 | 79.39±0.52 | 80.12±0.49 | |
| | *F1 Score* | **C** | 82.48±0.35 | 82.92±0.31 | 79.16±0.46 | 79.79±0.45 | 62.44 |
| | | **C+D** | 82.53±0.32 | 82.91±0.32 | 79.24±0.49 | 79.76±0.45 | |
| | | **C+N** | 82.51±0.36 | 83.20±0.31 | 79.23±0.47 | 80.11±0.49 | |
| | | **C+D+N** | 82.51±0.34 | **83.34**±0.30 | 79.19±0.43 | 80.04±0.43 | |

Table 4: Overall accuracy (mean and standard deviation, %) of joint model of mention type classifier and product classifier

by the golden labels. This assumption is validated in the following experiments.

## 5. Experiments

**Dataset** We use two datasets: *RLabel* for rule-labeled IDs (noisy labels) with 906,585 mentions, and *GLabel* for human-labeled IDs (golden labels) with 40,970 mentions. We split both datasets into training and test subsets at a ratio of 7 : 3, and extract 30% data from the training set for validation. The database contains $|\mathcal{P}| = 5{,}060$ products with $|\mathcal{B}| = 364$ brands, and the vocabulary size was $|\mathcal{V}| = 86{,}873$.

**Word Embeddings Pretraining** We apply the skip-gram model (Mikolov et al., 2013) to the corpus to obtain word embeddings. Since we collect the brand aliases as mentioned in §3.1., we average the embeddings of the brand aliases for each brand.

**Hyper-parameters** We use $D = 300$ dimensional pretrained word embeddings, and $d = 100$ dimensional vectors for LSTM and CNN hidden layers. The output embeddings of each encoder module and the product embeddings were also $D = 300$ dimensions. The CNN window size was 5. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001, mini-batches of size 32, and 10 epochs.

**Evaluation Setup** For each model, we train it on *RLabel*, *GLabel*, and *RLabel+GLabel* (§4.4., pretrain on *RLabel* and then train on *GLabel*), respectively, and then evaluated it on the *GLabel* testing dataset. We evaluate each model 10 times and compute the means and standard devi-

ations of the accuracies and F1 scores. Here we average the F1 scores weighted by their supports (the number of true instances for each label).

### 5.1. Baselines

Due to the lack of existing models that perfectly fit out special task HEL, we use two simple baselines — the heuristic rule (§3.4.) and the similarity classifier (described below). The similarity classifier computes the similarity of mentions with their candidate products. Note that the similarity classifier can be used only for product ID linking; that is, it can be used only as a baseline of product classification.

**Similarity Classifier** Given a mention $\mathtt{m} \in \mathcal{M}$, we select a candidate product $\mathfrak{p} \in \mathcal{P}$ with the same headword as the mention. We find a noun phrase $n$ by sentence parsing (Hsieh et al., 2007; Hsieh et al., 2012; Yang et al., 2008) the mention with the most significant Jaccard similarity coefficient[5] to the candidate, and compute the similarity of $\mathfrak{p}$ and $n$. Precisely, we propose two similarity methods — embeddings (denoted as *Emb*) and bag-of-words (denoted as *Bag*). First, we use the product name encoder (§4.3.3.) on both the product name $\mathfrak{p}$ and the noun phrase $n$, and compute the cosine similarity $s^{\mathfrak{p},n}_{\text{emb}}$ of those embeddings. We assign the ID of the candidate with the highest similarity to the mention. Another way is to replace the above embeddings by bags of words to compute $s^{\mathfrak{p},n}_{\text{bag}}$. Besides, we also evaluate the accuracy by adding the above similarities $s^{\mathfrak{p},n}_{\text{emb}} + s^{\mathfrak{p},n}_{\text{bag}}$ (denoted as *Emb+Bag*).

### 5.2. Evaluation of Mention Type Classifier

We evaluate the mention type classifier using the *RLabel*, *GLabel*, and *RLabel+GLabel* datasets, and compare the accuracies and F1 scores with the baselines (the heuristic rule, §3.4.). We also compute the F1 scores of the three mention types **`PID`**, **`OSP`**, and **`GP`**. From Tab. 2, we find that the model using *GLabel* outperforms those using other datasets. We conclude that the performance of the models affected by the heuristic rule (*RLabel* and *RLabel+GLabel*) does not meet our expectations, as the heuristic rule performs poorly on **`OSP`**.

### 5.3. Evaluation of Product Classifier

We test the product classifier with different encoder modules. The model using the context encoder (§4.3.1.) is denoted as **C**, that using the product description encoder (§4.3.2.) as **D**, and that using the product name encoder (§4.3.3.) as **N**. Since the context is necessary, we test all combinations with context — **C**, **C+D**, **C+N**, and **C+D+N**. We evaluate the above models using the *RLabel*, *GLabel*, and *RLabel+GLabel* datasets, and compare the accuracies and F1 scores with the baselines (the heuristic rule, §3.4., and the similarity classifiers, §5.1.). Here we use only the mentions labeled in **`PID`** for both training and testing.

Interestingly, the model using the *RLabel* dataset outperforms *RLabel* itself (the heuristic rule) by evaluating both accuracy and F-measure on *GLabel* even if we exclude the product databases (that is, model **C**, the model using the context encoder only). Since the heuristic rule uses a decision tree for labeling, we believe it is too arbitrary. However, the model produces embeddings for each product, uses similarity to determine the label, and avoids overdetermination.

### 5.4. Joint Evaluation of Mention Type Classifier and Product Classifier

Finally, we join the mention type classifier and product classifier for an end-to-end evaluation. We first apply the mention type classifier on all mentions in the testing data, and apply the product classifier on these labeled **`PID`** by mention type classifier to yield a specific ID. We further investigate the models using *GLabel* and *RLabel+GLabel* datasets only, as the performance of *RLabel* was relatively week. In Tab. 4, we compare the four combinations of both models with both datasets (*GLabel* and *RLabel+GLabel*).

We find that the model using *GLabel* on the mention type classifier and using *RLabel+GLabel* on the entity embeddings model achieves the best results. This is intuitive, as the mention type classifier and the entity embeddings model perform best with *GLabel* and *RLabel+GLabel*, respectively. Although the **C+N** model outperforms other encoder module combinations in terms of accuracy, the joint model using all encoder modules **C+D+N** yields the best F1 score.

## 6. Conclusion

The paper defines headword-oriented entity linking, a specialized entity linking problem in which only the headwords of the entities are to be linked to knowledge bases without named entity recognition. We create a labeled cosmetic corpus as a public benchmark for this problem, and propose a product-embedding model as a strong baseline to solve it, which simultaneously takes into account various types of context information and the information related to the products themselves. Moreover, we present a special transfer learning framework based on distant supervision. We believe HEL will fill many practical commercial needs, such as product recommendation and data mining for products. The model and the data set are also released to the public as the benchmark.

## 7. Acknowledgments

We are grateful for the insightful comments from anonymous reviewers. This work is supported by PIXNET Inc. and the Ministry of Science and Technology of Taiwan under grant numbers 109-2634-F-001-010.

## Bibliographical References

Agirre, E., Chang, A. X., Jurafsky, D., Manning, C. D., Spitkovsky, V. I., and Yeh, E. (2009). Stanford-UBC at TAC-KBP. In *TAC*.

Bunescu, R. and Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *11th conference of the European Chapter of the Association for Computational Linguistics*.

[5]The Jaccard similarity coefficient of two sets $A$ and $B$, also called the Jaccard Index, is defined as $|A \cap B|/|A \cup B|$.

Chen, Z. and Ji, H. (2011). Collaborative ranking: A case study on entity linking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 771–781. Association for Computational Linguistics.

Dredze, M., McNamee, P., Rao, D., Gerber, A., and Finin, T. (2010). Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Fonseca, E., Vanin, A., and Vieira, R. (2018). Nominal coreference resolution using semantic knowledge. In *International Conference on Computational Processing of the Portuguese Language*, pages 37–45. Springer.

Francis-Landau, M., Durrett, G., and Klein, D. (2016). Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734*.

Gupta, N., Singh, S., and Roth, D. (2017). Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hsieh, Y.-M., Yang, D.-C., and Chen, K.-J. (2007). Improve parsing performance by self-learning. *International Journal of Computational Linguistics & Chinese Language Processing*, 12(2):195–216.

Hsieh, Y.-M., Bai, M.-H., Chang, J. S., and Chen, K.-J. (2012). Improving PCFG chinese parsing with context-dependent probability re-estimation. In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 216–221.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kulkarni, S., Singh, A., Ramakrishnan, G., and Chakrabarti, S. (2009). Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM.

Lazic, N., Subramanya, A., Ringgaard, M., and Pereira, F. (2015). Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

Li, F., Zheng, Z., Bu, F., Tang, Y., Zhu, X., and Huang, M. (2009). THU QUANTA at TAC 2009 KBP and RTE track. In *TAC*.

Lin, T., Etzioni, O., et al. (2012). Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics.

Liu, X., Li, Y., Wu, H., Zhou, M., Wei, F., and Lu, Y. (2013). Entity linking for tweets. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1304–1311.

Ma, W.-Y. and Chen, K.-J. (2003a). A bottom-up merging algorithm for chinese unknown word extraction. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, volume 17, pages 31–38. Association for Computational Linguistics.

Ma, W.-Y. and Chen, K.-J. (2003b). Introduction to CKIP chinese word segmentation system for the first international chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, volume 17, pages 168–171. Association for Computational Linguistics.

McNamee, P., Dredze, M., Gerber, A., Garera, N., Finin, T., Mayfield, J., Piatko, C. D., Rao, D., Yarowsky, D., and Dreyer, M. (2009). HLTCOE approaches to knowledge base population at tac 2009. In *TAC*.

McNamee, P. (2010). HLTCOE efforts in entity linking at tac kbp 2010. In *TAC*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Shen, W., Wang, J., Luo, P., and Wang, M. (2012). Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM.

Shen, W., Wang, J., and Han, J. (2015). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Sun, Y., Lin, L., Tang, D., Yang, N., Ji, Z., and Wang, X. (2015). Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, pages 1333–1339.

Yamada, I., Shindo, H., Takeda, H., and Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

Yang, D.-C., Hsieh, Y.-M., and Chen, K.-J. (2008). Resolving ambiguities of chinese conjunctive structures by divide-and-conquer approaches. In *Proceedings of the Third International Joint Conference on Natural Language Processing*.

Zhang, W., Su, J., Tan, C. L., and Wang, W. T. (2010). Entity linking leveraging: automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1290–1298. Association for Computational Linguistics.

# A-HA: A Hybrid Approach for Hotel Recommendation

Kung-Hsiang, Huang
Rosetta.ai
steeve@rosetta.ai

Yi-Fu, Fu*
National Taiwan University
yifu.arlj@gmail.com

Yi-Ting, Lee*
National Taiwan University
amy19412000@gmail.com

Tzong-Hann, Lee
National Taiwan University
maxware0715@gmail.com

Yao-Chun, Chan
National Taiwan University
jan11781202@gmail.com

Yi-Hui, Lee
Univeristy of Texas, Dallas
yi-hui.lee@utdallas.edu

Shou-De, Lin
National Taiwan University
sdlin@csie.ntu.edu.tw

## ABSTRACT

Session-based recommender system refers to a specific type of recommender system that focuses more on the transactional structure of each session rather than the user and item interactions [16]. It is stated that the users' interactions are mostly homogeneous in the same sessions, while being heterogeneous across different sessions [5]. Therefore, it is essential to extract the interest dynamics of users within each session. The 2019 ACM Recsys Challenge [10] aims to apply session-based recommender systems to the domain of travel metasearch. The goal is to predict which hotels are clicked in the search results based on the context of each session. In this paper, we propose our approach to effectively tackle the challenge. It involves an ensemble of three models, LightGBM, XGBoost, and a Neural Network based on DeepFM [6] that is capable of handling sequential features. Our team, RosettaAI, won the 4th place in this challenge, scoring 0.679933 on the final leaderboard. The source code is available online [1].

## CCS CONCEPTS

• **Information systems → Recommender systems**.

## KEYWORDS

Hotel recommendation, Gradient boosting machine, Neural networks

**ACM Reference Format:**
Kung-Hsiang, Huang, Yi-Fu, Fu, Yi-Ting, Lee, Tzong-Hann, Lee, Yao-Chun, Chan, Yi-Hui, Lee, and Shou-De, Lin. 2019. A-HA: A Hybrid Approach for Hotel Recommendation . In *Proceedings of the ACM Recommender Systems Challenge 2019 Workshop (RecSys Challenge '19), September 20, 2019, Copenhagen, Denmark*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3359555.3359560

*Both authors contributed equally to the paper
[1]https://github.com/rosetta-ai/rosetta_recsys2019


*RecSys Challenge '19, September 20, 2019, Copenhagen, Denmark*

ACM ISBN 978-1-4503-7667-9/19/09…$15.00
https://doi.org/10.1145/3359555.3359560

## 1 INTRODUCTION

With the ever-growing number of available accommodations online, successfully matching hotels with the user's interest becomes gradually important. Since the majority of users visiting these sites are not logged-in, in addition to considering the users' past behaviors, an effectual recommender system must be able to model the transition of user interests in the sessions. Therefore, to tackle the challenge, we need to build a session-based and context-aware recommender system tailored for hotel recommendation.

Several studies [14] [7] have attempted to solve this task with Collaborative Filtering (CF) [13]. Essentially, this algorithm constructs a user-item matrix based on the past interactions of all users. Predictions are made by computing the similarity between items or users. Although CF successfully extracts the latent patterns behind user and item interactions, it suffers from two main drawbacks. First, it ignores the contextual feature of each session. Such information is critical for modeling the dynamics of user interest throughout the session. In addition, the performance of CF decreases significantly when the sparsity of the user-item matrix is high. This holds in this challenge. The user-item matrix is highly sparse due to the large portion of non-logged-in users.

Our team leveraged an ensemble of three different models, Neural Network, LightGBM, and XGBoost to solve the challenge. We utilized the Neural Network's expressive ability in modeling sequential data with Bi-directional Gated Recurrent Units (Bi-GRUs) [3], while using the other two models to learn from structural data efficiently. By ensembling these three models, we can take advantage of the strengths of each of them.

In this study, we discovered that although unable to process sequential features, LightGBM and XGBoost still outperform Neural Network significantly thanks to their efficacy in extracting information from tabular data. Furthermore, session-based features, especially those related to the immediate last interaction, are found to be the most important features in model's performance.

The rest of the paper is organized as follows. First, the framework of the challenge will be described. Then, we will explain our approach to the challenge, including the loss function, the models,

the ensemble structure, and the feature engineering. Finally, the experiments and results will be illustrated.

## 2 CHALLENGE TASK

### 2.1 Problem Definition

In the 2019 ACM Recsys Challenge, trivago provided a dataset of browse logs on the Trivago website, which consists of consecutive actions of users and referenced items (e.g. hotel). There are various action types, including *interaction with item image*, *change of sort order, filter selection*, *search for destination*, *clickout item*, and etc. The goal of the competition is to predict the exact clicked item among at most 25 candidates in the rows associated with the *clickout item* action, given the information of series of actions right before the *clickout item* action occurs. The evaluation metric is Mean Reciprocal Rank (MRR), defined as follows [12]:

$$MRR = \frac{1}{|Q|}\sum_{i=0}^{|Q|}\frac{1}{rank_i} \tag{1}$$

where Q denotes the total number of samples, and $rank_i$ denotes the rank of the first correct answer for sample i.

### 2.2 Data Description

As stated in the Challenge Dataset webpage, the provided dataset contains training data, test data, and metadata of accommodations (items). In particular, there are 730803 users, 400277 items, and 910683 independent sessions, each of which is composed of series of actions involving one user and several items. There may exist one or more *clickout item* actions in a single session. In a *clickout item* action, all possible accommodations and their prices are listed in the same order as they were displayed to the user, also known as the *impressions list* and *price list*.

## 3 APPROACH

In this section, we will describe our approach to the challenge. We first define the loss function as well as the data preprocessing pipelines. Then, two different types of models we implemented, Neural Network, and Gradient Boosting Machine are illustrated. Lastly, we demonstrates the important features we engineered.

### 3.1 Loss Function

Inspired by this study [4], we adopted binary cross-entropy (BCE) loss as our loss function. Each of the item in the impression is broken down into individual samples. The labels associated with the clicked item are 1, while the others being 0. Mathematically, the loss function is defined as follows:

$$L = \sum_{i=0}^{N} y_i log(\hat{y_i}) + (1 - y_i) log((1 - \hat{y_i})) \tag{2}$$

where N denotes the total number of broken down samples, $y_i \in [0,1]$ denotes the ground truth, and $\hat{y_i}$ denotes the output of the model, where $\{\hat{y_i} \in \mathbb{R}; 0 \leq \hat{y_i} \leq 1\}$.

### 3.2 Preprocessing

The preprocessing procedure involves three steps, removing invalid *clickout item* rows, breaking down impression, and encoding categorical feature. In the first step, a *clickout item* row is considered invalid if its reference value does not present in its impression. Such rows containing no positive ground truth are removed from the training and validation sets. Then, as aforementioned, the impression of each row is broken down to $\mathcal{I}_i$ samples, where $\mathcal{I}_i$ is the number of items in the i-th impression. Categorical features are encoded from 0 to $C_j - 1$, where $C_j$ denotes the number of unique value for the j-th categorical feature. In addition to saving memory, such encoding was chosen over one-hot encoding for later feeding the encoded features into embedding layers efficiently.

### 3.3 Neural Network

Motivated by DeepFM [6], we propose a Neural Network that is capable of modeling the second-order feature interaction efficiently, while taking into account the temporal dynamics of user-item interactions at the same time. As depicted in Figure 1, the input features can be divided into three parts, continuous features, categorical features, and sequential categorical features. The following sub sections will illustrate the detail of each of them with the notation below.

- D: 1-dimensional continuous features. $D_i$ denotes the i-th continuous feature.
- E: 1-dimensional encoded categorical features. $E_i$ denotes the i-th categorical feature.
- F: 2-dimensional encoded sequential categorical features. $F_i$ denotes the i-th sequential categorical feature.

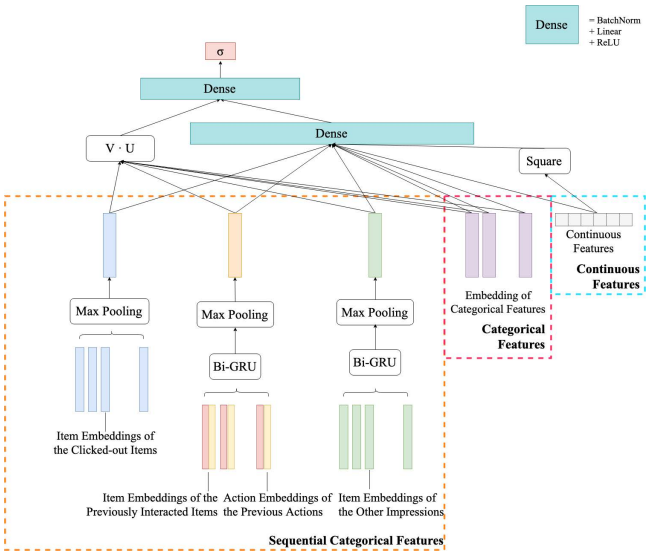



**Figure 1: The model architecture of the proposed Neural Network. The inputs constitutes of three parts, continuous features, categorical features, and sequential categorical features. All layers are densely connected.**

*3.3.1 Continuous Features.* Continuous features must be properly normalized for Neural Networks to learn well. Two different normalization techniques were applied to D, uniform normalization to [0, 1] and the Rank Gauss transformation [15]. The latter one ranks the continuous values for each $D_i$, uniformly maps the ranking onto [0, 1], and transforms this with the inverse error function. The error function, which is also known as the Gauss error function, is mathematically defined as [1]:

$$erf(x) = \int_0^x e^{-t^2} dt \tag{3}$$

The two methods work the best compared with standardization. Thus, both of them are used for normalization.

*3.3.2 Categorical Features & Sequential Categorical Features.* Both E and F are embedded to obtain a u-dimensional representation for each feature, where u $\in \mathbb{N}$ . Notably, features under the same ID space share the same embedding weights. For instance, there exists a single embedding of item IDs that distinct features from E and F may contain (item IDs of the impression and the IDs of the items that the user had interacted in the current session).

To model the dynamics of user interest throughout the session, the embeddings of item IDs and actions are concatenated, and processed by a Bi-directional GRU. The outputs of GRU for each time step are then aggregated with max pooling. Formally, it can be described as:

$$G = g([F_{item}, F_{action}], \theta)$$
$$V = maxpool(G)$$

where $\theta$ denotes the parameters of Bi-GRU, g denotes the Bi-GRU transformation, G denotes the output of Bi-GRU, and V denotes the aggregated vector. Similarly, the other features in F boils down to u-dimensional vectors.

*3.3.3 Feature Interaction.* As stated in [6], feature interaction allows the model to discover the implicit meaning behind user's behavior. Therefore, a proper feature interaction mechanism is important for improving the performance of the model. DeepFM computes the Hadamard products of the embedding vectors of each pair of features, and aggregating them with weight-1 connections. Essentially, it is equivalent to summing the element-wise products of the embeddings. This can be achieved efficiently by computing difference between the square of sum and the sum of square of these embedding vectors. Let $V_i, V_j, V_k \in \mathbb{R}^u$ denotes the embedding vectors for feature i, j, and k, we can derive the following:

$$(V_i + V_j + V_k)^2 = V_i^2 + V_j^2 + V_k^2 + 2(V_i \circ V_j + V_j \circ V_k + V_i \circ V_k)$$
$$(V_i \circ V_j + V_j \circ V_k + V_i \circ V_k) = \frac{1}{2}((V_i + V_j + V_k)^2 - (V_i^2 + V_j^2 + V_k^2))$$

where $\circ$ denotes the Hadamard product operation.

Hence, it is not necessary to explicitly compute the Hadamard products for each pair of embedding vectors; instead, we simply compute the two terms on the right-hand-side in the above equation.

## 3.4 Gradient Boosting Machine

Tree-based models usually perform the best in structured data, especially for gradient boosting machines. These models have been dominant in Kaggle competitions involving tabular data. We selected LightGBM [8] and XGBoost [2] to increase model diversity for ensemble. These two models takes all the features fed to the Neural Network, except for the sequential categorical features, with additional hand-crafted features. It turns out that both of these two models outperform the Neural Network by a large margin. Details of the experiment results will be provided in the latter section.

## 3.5 Ensemble

Our ensemble method is an weighted average of the predictions for each of the three models discussed above. Specifically, the predictions are blended with the following ratio:

$$NeuralNetwork : LightGBM : XGBoost = 1 : 7 : 4$$

Empirically, this ratio generates the best results.

## 3.6 Feature Engineering

We conducted feature engineering with the aim of capturing all different aspects. The different set of features are described by the following subsections:

*3.6.1 Impression-based features.* There are some obvious information of the impression list including price, city, platform, rating, and star of those items on impression list. In addition, we engineered some hidden messages such as the rank of each item's price, the position of the item on the impression list, and the length of the impression list.

*3.6.2 Session-based features.*

- Time difference feature: The time difference between current and the last step action.
- Item time difference: The time difference between when an item was last interacted and when the clickout item action took place.
- Last interacted item impression index: The position of the last interacted item in the current impression list.
- Equal last interacted item: Whether this item in impression list equal to the last interacted item in the current session.
- Local count feature: For each section, we compute the count of different types of interaction for each item in the impression list.
- Last interact index: The impression index of the the last interacted item in the current session.
- Predicted next impression index: To model the eye movement of the user on the website, we leveraged the position of the last interacted item and the second last interacted item on the current impression list. Specifically, this feature is computed as follows: `Predicted next impression index = last impression index + ( last impression index - second last impression index )`, where the second term calculates the movement of the last interaction.

*3.6.3 Aggregation features.*

- Target encoding: To better approximate the priors, we aggregate the mean value for some of the categorical features, such as price rank and impression index.
- City price bin: Considering the different price indices of different cities, we categories prices into bins based on city.

# 4 EXPERIMENTS

## 4.1 Experimental Settings

To prevent the models from seeing the future clickout events, which might lead to label leakage, we adopted the *leave-one-out* evaluation as our validation strategy. In specific, 50000 sessions were randomly selected, only the last clickout events of which are used as validation data. The validation scores shown in this paper refer to the models' performance on this validation set.

## 4.2 Training Process

The Neural Network was implemented in PyTorch [11]. Adam [9] was used for optimizing the Neural Network with learning rate=0.001 and weight decay=0. To boost the training speed, the batch size was set to 1024. We train the Neural Network only for 1 epoch on a single Nvidia Tesla M60 GPU. In our experiments, the Neural Network usually starts to overfit when trained beyond 1 epoch. Any regularization technique other than batch normalization, such as dropout and l2-regularization, only result in poorer performance. All the weights are randomly initialized, following the default setting of PyTorch.

As for LightGBM and XGBoost, the learning rate was set to 0.01 and 0.02, respectively. The other training parameters are in general the same. Their maximum number of boosting rounds and early stopping rounds are set to 50000 and 500. BCE loss is used as a proxy for their early stopping criteria since computing MRR for every round during training is computationally expensive. Their base learners are gradient boosting decision trees, which perform the best empirically. They are all trained on 48 CPU cores.

## 4.3 Experimental Results

The computation time and performance of the three models are evaluated. As demonstrated in Table 1, the Neural Network takes the least time to train as it converge the fastest. LightGBM requires the longest training time since it stopped at its maximum boosting rounds, while XGBoost stopped at less than 20000 rounds. The long training time for LightGBM also explains its long inference time. The longer the model is trained, the more sophisticated the model; thus, taking more time while performing prediction.

Comparing the three single models, tree-based models outperform the Neural Network by a large margin, as shown in Table 2, even though sequential features are not fed into these models. We hypothesize that this is because tree-based models are efficient in extracting information from tabular data. In addition, such data does not exist evident hierarchical structure compared with image or text data, where Neural Networks dominates with their strong hierarchical representation abilities. Among these three, LightGBM performs the best in local validation. However, the best LightGBM prediction was not submitted since it was produced almost at the end of the challenge when we do not have many submission opportunities. That prediction result was only used for ensemble. By blending the prediction of three models with the ratio mentioned previously, the ensemble achieved the best MRR on the public Leaderboard.

| | NN | LightGBM | XGBoost |
|---|---|---|---|
| Training Time | **6** | 14 | 10 |
| Inference Time | **0.3** | 1.2 | 1 |

**Table 1: Computation time comparison. (Hours)**

| | NN | LightGBM | XGBoost | Ensemble |
|---|---|---|---|---|
| Validation | 0.675206 | **0.685787** | 0.684521 | N/A |
| Leaderboard | 0.672117 | N/A | 0.681128 | **0.682128** |

**Table 2: Performance comparison. (MRR)**

| Feature Name | Importance Score |
|---|---|
| Equal Last Interacted Item | $8.9 \times 10^7$ |
| Item Time Difference | $3.9 \times 10^7$ |
| Impression Index | $3.2 \times 10^7$ |
| Last Interact Index | $2.3 \times 10^7$ |
| Time Difference | $1.1 \times 10^7$ |

**Table 3: Top 5 important features for LightGBM**

Table 3 shows the top 5 important features generated by our best LightGBM model. It seems that session-based features related to the immediate last action before the *clickout item* actions play the most important roles in improving our modes. For instance, *Equal Last Interacted Item* indicates if an item in the impression list is the same as the immediate last interacted item, while *Item Time Difference* refers to the time difference between when an item was last interacted and when the *clickout item* action took place.

# 5 CONCLUSION

In this paper, we described our approach to the 2019 ACM RecSys Challenge. The challenge was formulated as a binary classification problem where BCE loss was adopted to optimize the models. An ensemble of three models were presented to tackle the challenge, including Neural Network, LightGBM and XGBoost. We discovered that tree-based models are more effective in extracting information from tabular data than Neural Networks in this challenge. Moreover, among all the features, session-based features associated with the immediate last interaction are the most critical in terms of improving the evaluation metric. In the end, our team won the fourth place on the final leaderboard of this challenge, suggesting that our proposed ensemble model is effective in solving such task.

# REFERENCES

[1] Larry C Andrews and Larry C Andrews. 1992. *Special functions of mathematics for engineers*. McGraw-Hill New York.
[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785
[3] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

*CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1905.06482* (2019).

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[7] Ya-Han Hu, Pei-Ju Lee, Kuanchin Chen, J Michael Tarn, and Duyen-Vi Dang. 2016. Hotel Recommendation System based on Review and Context Information: a Collaborative Filtering Appro.. In *PACIS*. 221.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. 2019. RecSys Challenge 2019: Session-based Hotel Recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems (RecSys '19)*. ACM, New York, NY, USA, 2. https://doi.org/10.1145/3298689.3346974

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[12] Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems.. In *LREC*.

[13] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* 1 (2001), 285–295.

[14] Qusai Shambour, Mouath Hourani, and Salam Fraihat. 2016. An item-based multi-criteria collaborative filtering algorithm for personalized recommender systems. *International Journal of Advanced Computer Science and Applications* 7, 8 (2016), 274–279.

[15] Tri Duc Nguyen Tang. 2019. Knowledge Distillation with NN + RankGauss. Retrieved June 5, 2019 from https://www.kaggle.com/mathormad/knowledge-distillation-with-nn-rankgauss/data

[16] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).

# Conditional Relationship Extraction for Diseases and Symptoms by a Web Search-based Approach

1st Yi-Hui Lee*
*Department of Computer Science*
*The University of Texas at Dallas*
Richardson, Texas
yi-hui.lee@utdallas.edu

2nd Jia-Ling Koh
*Department of Computer Science and Information Engineering*
*National Taiwan Normal University*
Taipei, Taiwan
jlkoh@csie.ntnu.edu.tw

***Abstract*—This paper studies the strategies of automatically extracting the conditional relationships between diseases and symptoms from a Chinese encyclopedia site and the disease-related web pages searched from the Internet. At first, the seed symptoms of a disease are extracted from an online medical encyclopedia automatically. These seed symptoms are utilized as query keywords to automatically find more symptoms of a disease from the unstructured documents of the disease-related search results. Next, a jointly learning method is used to construct the embedded representations of the conditional terms and pattern terms. Besides, the semantic similarity matrix of conditional terms is computed through the co-clustering algorithm to discover the representative conditional terms of the clusters. The result of experiments shows that the proposed method, which discovers the semantically related symptoms of a disease associated with conditionals, achieves high accuracy. Besides, many unusually known symptoms considered by the medical experts are discovered, which may be noticeable symptoms needing further verification in the future.**

***Index Terms*—Text Mining, Information Extraction, Semantic Networks**

## I. Introduction

In recent years, medical data mining has become an important research issue. More and more different data mining tasks in healthcare are studied. Most of the analytics in healthcare today focus on structured data, such as the Electronic Health Records (EHRs) are used to assist doctors in the decision making of treatments [1]. Furthermore, the track on Clinical Decision Support in the Text REtrieval Conference (TREC-CDS) [2] aimed to retrieve the diagnosis, order, and treatment from medical records to answer the medical questions. On the other hand, the unstructured healthcare data collected from the public posting on the social media platforms can be hard to manage but provide possibly useful information. For example, [3] studied how to predict the drug reactions from discussion forums, [4] detected adverse drug events in Tweets with semi-supervised convolutional neural networks, and [5] created a catalogue of real-world treatments from online Autism communities.

To build a knowledge base from the unstructured medical data extremely improves the usage and understanding of the healthcare documents. A knowledge base not only can provide answers automatically for the Question Answering (QA) systems [6]–[8] but also help semantics understanding for natural language processing [9]. A lot of tasks have been proposed to build general-purpose knowledge bases from different corpora automatically. For example, the online encyclopedia is considered to be a good resource to construct a knowledge base, such as the DBpedia[1] constructed from Wikipedia[2] [10] and the NELL [11]–[13] learned knowledge from the web.

In order to automatically process and analyze the Electronic Medical Records (EMRs) to provide a clinical decision support system, it is a critical step to retrieve the relationships between medical concepts, especially for diseases and symptoms. However, most medical knowledge bases are in English. [14] provided an approach to extract symptoms and symptom-related entities from healthcare websites and encyclopedia sites for constructing a medical knowledge base in Chinese. However, various symptoms of a disease occur under different conditions. For each disease, it is necessary to discover the specific conditional terms for the corresponding symptoms. For example, `chest pain` is a symptom of `lung cancer` in the `early stage`. Then the `has_symptom` relationship between `lung cancer` and `chest pain` should have the corresponding conditional term `early`. Accordingly, in this paper, we aimed to automatically extract the conditional relationship between diseases and symptoms from an encyclopedia site and healthcare websites in Chinese. By giving a disease, say `lung cancer`, the symptoms of `lung cancer` will be extracted associated with the conditions as triples, such as `has_symptom (lung cancer, early, chest pain)`.

The challenges of this task are as follows:

- The information provided in the Chinese medical encyclopedia is limited. Only a few symptoms of a disease can be extracted.
- It is not trivial to extract the symptoms related to a given disease from the healthcare web pages because the information is unstructured.
- It is difficult to decide the useful conditions from the contexts of a symptom.

*Work performed while this author was at NTNU.

[1] http://wiki.dbpedia.org
[2] https://www.wikipedia.org

978-1-5386-7325-6/18/$31.00 ©2018 IEEE
DOI 10.1109/WI.2018.00-38

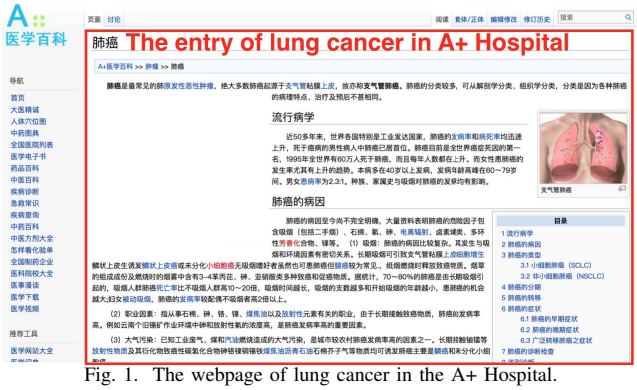


Fig. 1. The webpage of lung cancer in the A+ Hospital.

Inspired by the idea of constructing DBpedia from Wikipedia, the first step of our work is to discover the relationships between diseases and symptoms according to the hyperlinks in the web pages of the A+ hospital[3]. The A+ hospital is a Wikipedia-like website in Chinese, which provides lots of healthcare information of diseases and symptoms as shown in Fig. 1. Next, to complete the symptoms of a given disease, we utilized the web search results, where many web pages provide healthcare information of diseases. How to extract possible conditional symptoms from the noisy web search results is the main issue studied in this paper.

We provide an off-line conditional relationship constructing system composed the following three modules: (1) Seed symptoms extraction: data in A+ hospital was used to initially construct the relationship between a disease and its seed symptoms, as shown in Table I. (2) Extended symptoms discovery: the disease and the seed symptoms are used as query keywords to perform web search for finding the other possible symptoms of the disease. The scoring method is proposed to select the top $k$ candidate symptoms to form the extended symptoms as shown in Table I. (3) Conditional term discovery: the contexts of the symptom terms in the search results are collected as the candidates of conditional terms. A jointly learning approach is performed to construct the embedded representation for the conditional terms and pattern terms. Then the K-means plus plus algorithm is used to cluster the candidate conditional terms according to the similarity measure of their embedded representations. Finally, the most frequent terms in each cluster are chosen as the representative conditional terms to generate the `has_symptom` relationships with conditions.

The contributions of this work are summarized as follows:

- We provided a measuring method to rank the candidate symptoms of a disease extracted from the unstructured documents of the disease-related search results.
- An embedded representation learning method for each candidate conditional term was designed in an unsupervised approach.
- We applied the co-clustering approach to discover the representative conditional terms.
- We provided a framework to construct the conditional relationship of disease and symptom in Chinese.

[3]http://www.a-hospital.com

TABLE I
THE HAS_SYMPTOM RELATIONSHIP FOR LUNG CANCER AND THE SYMPTOMS.

| Seed Symptom | Extended Symptoms |
|---|---|
| 胸痛(chest pain) | 脱水(dehydration) |
| 盗汗(night sweating) | 谵妄(delirium) |
| 寒战(shivering) | 痴呆(dementia) |
| 头痛(headache) | 脓痰(purulent sputum) |
| 意识障碍(disturbance of consciousness) | 大小便失禁(incontinence) |
| 瘫痪(paralysis) | 视物模糊(blurring of vision) |
| 低热(low fever) | 气促(anhelation) |
| 咳血(hemoptysis) | 肝转移(liver metastases) |
| 压痛(tenderness) | 溃疡(ulcer) |
| 心悸(palpitation) | 干呕(retching) |
| 抽搐(tic disorder) | 黑蒙(amaurosis fugax) |
| and 15 other mores | and 89 other mores |

The rest of the paper is organized as follows. After a summary of related works in Section II, we introduce the proposed methods for extracting the relationship between a disease and the symptoms in Section III. In Section IV, we present the strategies for conditional terms discovery. Section V describes the evaluation of the proposed methods. Section VI concludes this paper and gives the future work.

## II. Related Work

Many data mining tasks on the medical domain were studied, such as Sun et al. [1] developed a framework, which automatically recommends treatments to doctors. Feldman et al. [3] proposed an approach to predict adverse drug reactions prior to the Food and Drug Administration (FDA). This paper proposed a text mining methodology from four online medical message boards to construct the drug-symptom relationships. Then, the lift measure is utilized to evaluate the correlation between drugs and adverse drug reactions.

Furthermore, building a knowledge base from the unstructured medical data extremely improves the understanding of the medical records. Goodwin et al. [6] presented a novel framework to answer medical questions by retrieving relevant medical articles, which is a challenge of TREC-CDS [2]. By retrieving the diagnosis, orders, and treatments from medical records, the proposed framework built a probabilistic knowledge graph: a clinical picture and therapy graph from a large collection of EMRs. Then the probabilistic inference strategy was applied to identify the answers for selecting and ranking the scientific articles containing the answers. For solving the same problem, [15] used MetaMap, a medical concept recognizer, to extract medical concepts. Besides, a Wikipedia knowledge base was used to predict the patient diagnosis. Accordingly, the original query is expanded with the predicted diagnosis to search relevant articles.

Many researches have proposed an impressive result in building general-purpose knowledge bases automatically, such as NELL [11]–[13] learning knowledge from the web by bootstrapping strategy and DBpedia constructed from Wikipedia [10]. NELL used a pattern-based strategy to build a knowledge base. The DBpedia project built a large-scale, multilingual knowledge base by extracting structured data from Wikipedia [10]. On the other hand, Wang et al. [16] extracted the

concepts and the instances from Hudong, which is a Chinese encyclopedia. A method was proposed to learn ontology from the category system and Infobox schema in Hudong. Based on the ontology, the instances were extracted accordingly. Furthermore, Li et al. [17] built a cross-lingual knowledge base to integrate four wikis of different languages.

Although many methods have been developed to build knowledge bases automatically from the encyclopedia, it is possible that the information provided in the encyclopedia is not complete. For solving this problem, Savenkov and Agichtein [7] provided a Text2KB system to translate a natural language question to the Knowledge Base (KB) entities and predicates. The system utilized textual data from web search results, community question answering platforms, and a general text document collection. The topic entities in a question were detected and the question phrases were mapped to predicates in knowledge bases. West et al. [8] built an end-to-end pipeline for knowledge base completion based on search-based question answering. They used a question-answering system to retrieve relevant and up-to-date text passages in order to extract the candidate answers linking to the Freebase entities. Zhang et al. [14] constructed a knowledge base of symptoms automatically from eight healthcare websites, three Chinese encyclopedia sites, and symptoms extracted from EMRs. The categories of encyclopedia sites were used to extract target entities to train a classifier for deciding entity types. Besides, the duplications and inconsistencies in different resources were considered.

Wang et al. [18] showed that extracting the condition of a question is useful to solve the question answering problems based on a knowledge base. To extract the conditional knowledge from the dialogues, the condition terms are extracted by a bootstrapped pattern-learning method. Then the condition embedding model and the pattern embedding model are built by a supervised learning paradigm. Moreover, a new objective function is designed to modify the skip-gram model to the jointly embedding model. After that, the word embedding of conditions and patterns are utilized for co-clustering and discover the representative conditions. This paper provided us the innovative idea that condition is significant when describing a relationship between a disease and its symptoms.

## III. Relation Extraction

### A. Relation Extraction Problem

In our task, there are two kinds of input sources for extracting the relationships between diseases and symptoms: one is the website of A+ hospital and the other one is the non-structured documents of web search results. The goal is to find the triple *(d, c, s)*, which denotes a disease *d* has symptom *s* under the condition *c*.

The proposed approach consists of three parts of processing modules as shown in Fig. 2: (1) Seed symptoms extraction, (2) Extended Symptoms discovery, and (3) Conditional terms discovery. The details of the modules are explained in Section III-B, III-C, and Section IV, respectively.

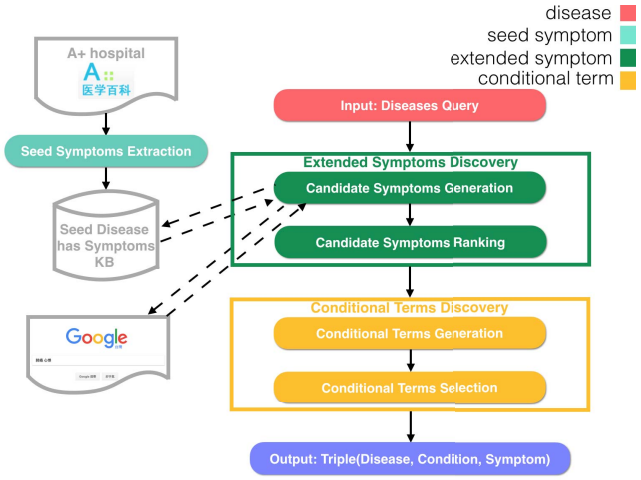



Fig. 2. The system architecture.


Fig. 3. An example of seed symptoms extraction.

### B. Seed Symptoms Extraction

In order to extract the seed symptoms of diseases from the A+ hospital automatically, the scrapy[4] toolkit was applied to scrape all the web pages describing the diseases and symptoms. Then a Chinese dictionary of the diseases and symptoms is constructed accordingly. Besides, in the entry of a disease *d*, the mentioned terms with a hyperlink to the category `symptom` are selected to be the seed symptoms of *d* as shown in Fig. 3.

### C. Extended Symptoms Discovery

A disease and its every seed symptom are used as the query keywords to search related web pages from the Internet. According to the web pages returned from the search engine, the paragraphs which contain the term 症状, which is the term of `symptom` in Chinese, are retrieved for further processing.

For example, when the disease is 肺癌(`lung cancer`) and the seed symptom is 胸痛(`chest pain`), the query keywords given to the search engine are 肺癌(`lung cancer`) and 胸痛(`chest pain`). One of the retrieved paragraphs is shown in Fig. 4. The terms marked in light green are seed symptoms. Besides, the other terms marked in dark green are the candidate symptoms, which are in the dictionary of symptoms.

[4]https://scrapy.org

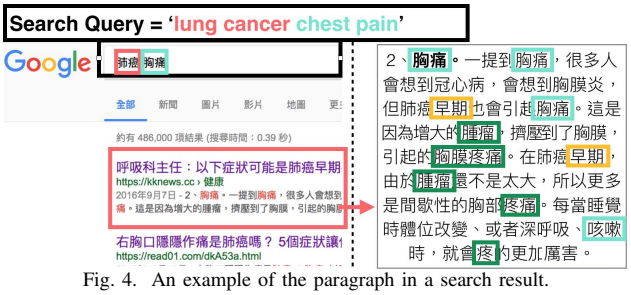


Fig. 4. An example of the paragraph in a search result.

In order to score the candidate symptoms, the proposed idea is that the more a seed symptom is important to the disease and a candidate symptom is related to the seed symptom, the more possible that the candidate symptom is highly related to the disease. Accordingly, the scoring function of a candidate symptom $C_i$ consists of two parts: (1) the significance of a seed symptom $S_j$, i.e. $ImportantScore(S_j)$, and (2) the related weight of the candidate symptom $C_i$ to the seed symptom $S_j$, i.e. $RelatedWeight(C_i, S_j)$. The defined equation is shown as below:

$$Score(C_i) = \sum_{S_j \in Seed} RelatedWeight(C_i, S_j) \times ImportantScore(S_j) \quad (1)$$

Three different strategies are designed to compute the important score of a seed symptom. The first twos use a graph model to represent the relationship among the seed symptoms, where the edges are weighted with two different methods. Then the random walk with restart paradigm [19] is used to evaluate the significant score of the seed symptoms. The third one evaluates the centrality of a seed symptom according to the average similarity measure with the other seed symptoms. The details of the three strategies are described as follows:

- Random walk with co-occurred times as edge weight ($RW_C$): A graph representing the relationships among the seed symptoms is constructed, where the vertices correspond to the seed symptoms. Besides, the edge weight between each pair of symptoms is assigned the co-occurred frequency when both symptoms appear in the same paragraphs of the search results within a window size of 10. After performing the PageRank algorithm [20] to determine and estimate how important each seed symptom is, which is called the representative scores. The representative scores of the seed symptoms are normalized by the maximum representative scores to get $ImportantScore(S_j)$ for each seed symptom $S_j$ as shown in Table II. In order to score the candidate symptoms based on their relatedness with the high representative symptoms, only the seed symptoms whose important score higher than the given threshold value 0.1 are selected into $Seed$.
- Random walk with W2V cosine similarity as edge weight $RW_{W2V}$: According to the graph modeling the relationships among the seed symptoms, the PageRank algorithm is used to compute $ImportantScore(S_j)$ for each seed symptom $S_j$. Here the edge weight between two symptoms is set to be the cosine similarity of the word embedding of the symptoms. By collecting the disease-related web pages as the training corpus, the word embedding for the seed symptoms are learned from the word2vector (W2V) skip-gram model [21][5], which are provided in the gensim[6] toolkit, with window size = 10 and dimension = 250. The seed symptoms whose important scores higher than the given threshold value 0.1 are selected into $Seed$.
- Word to vector cosine similarity Average $W2V_{avg}$: According to the learned word embedding of the seed symptoms, for each seed symptom $S_j$, in the $W2V_{avg}$ method, $ImportantScore(S_j)$ is the average cosine similarity of $S_j$ with the other seed symptoms. Besides, $Seed$ remains all the seed symptoms.

TABLE II
AN EXAMPLE SHOWING THE OBTAINED IMPORTANT SCORES OF THE SEED SYMPTOMS AFTER NORMALIZATION.

| (score) | *s1*: (chest pain) | *s2*: (cough) | *s3*: (malnutrition) | *s4*: (hemoptysis) |
|---|---|---|---|---|
| **Representative** | 1.117 | 1.0 | 0.95 | 0.933 |
| **Important** | 1.0 | 0.8953 | 0.8505 | 0.8353 |

Furthermore, the relative weight of a candidate symptom $C_i$ to seed symptom $S_j$, denoted by $RelatedWeight(C_i, S_j)$, is computed by the following two methods.

- Conditional probability $(CP)$: $RelatedWeight(C_i, S_j)$ is set to be the conditional probability $P(C_i|S_j)$, which is computed by dividing the co-occurrence frequency of $C_i$ and $S_j$ to the frequency of $S_j$ in the search results.
- Word to vector semantic cosine similarity $W2V_{sim}$: $RelatedWeight(C_i, S_j)$ is set to be the cosine similarity $cos_{sim}(C_i, S_j)$, where $C_i$ and $S_j$ are represented by their word embedding by word2vec.

Among the candidate symptoms, the symptoms with the top $k$ $Score(C_i)$ values are selected further to find their conditional terms.

## IV. Conditional Term Discovery

In the conditional term discovery module, the processing consists of the following two steps: conditional terms generation and conditional terms selection.

### A. Conditional Terms Generation

Based on the best performance approach in the extended symptoms discovery stage (which will be carefully discussed in Section V-B), the top $k$ ($k$=100) representative symptoms are selected. The conditional terms are the ones appearing nearby the symptoms in the disease-related web pages. Therefore, the contexts appearing within a window size of 10 with the representative symptoms in the search results are collected to form the set of candidate conditional terms. In order to remove the noisy terms, only the following three types of conditional terms are remaining:

- Temporal terms: the terms whose POS tagging[7] are about time, which are denoted by $C_t$.

[5]Skip-gram is a model architecture to compute continuous vector representations of words from a very large data set. It tries to maximize the predicting probability of words within a certain range before and after the target word.
[6]https://radimrehurek.com/gensim/models/word2vec.html
[7]POS tags (Part-of-Speech tags) are special labels assigned to each token (word) in a sentence to indicate their syntax properties, where the POS tags t and td are about time, the POS tags adv., adj., v., and n. are about status.

- Status terms: the terms whose POS tagging are about status, which are denoted by $C_s$.
- Organ terms: the terms defined in the organ' s pages of the A+ hospital, which are denoted by $C_o$.

In the following, these three different types of candidate conditional terms are processed separately.

### B. Conditional Terms Selection

In order to discover the groups of conditional terms with similar semantics, the embedded representation for each candidate conditional term is unsupervised learned firstly. For each type of conditional terms, says $C_t$, the co-occurred conditional terms belonging to the other two types of conditional terms, $C_s$ and $C_o$, are called their pattern terms. It is assumed that the semantics of a conditional term is related to the co-occurred symptoms, conditional terms, and the pattern terms in the same paragraph. Accordingly, we apply the skip-gram model on the set of created documents, in which each contains the symptoms, conditional terms, and the patterns terms appearing together in the same disease-related search result. For example, for the conditional terms in $C_t$, a created document in the training data consists of the mentioned symptoms: `chest pain` and `cough`; the co-occurred conditional terms in $C_t$: `often` and `early`; and the pattern terms: `close to` and `heart`.

Next, the co-clustering algorithm is performed to generate the similarity matrix between candidate conditional terms. Three matrices are obtained by computing the similarity of the embedded representations between pairs of conditional terms, a conditional term and a pattern term, and pairs of pattern terms. These matrices are called the Condition-Condition similarity matrix denoted as $SC$ with size $m \times m$, the Condition-Pattern similarity matrix demoted as $M$ with size $n \times m$, and the Pattern-Pattern similarity matrix denoted as $SR$ with size $n \times n$, respectively. In the matrix $M$, $m_{ij}$ corresponds to the cosine similarity between the embedded representations of the $i^{th}$ pattern term and the $j^{th}$ conditional term, respectively.

It is assumed that the more two conditional terms are related to the semantically related pattern terms, the more these two conditional terms are related. Similarly, the more two pattern terms are related to the semantically related conditional terms, the more these two pattern terms are related. Accordingly, the Condition-Condition cosine similarity matrix is updated by the co-clustering approach iteratively.

The embedding based co-clustering is performed based on the algorithm described in Bisson et al. [22] and Wang et al. [18]. At each iteration $t$, the new similarity matrix $SR_t$ is computed by using the similarity matrix $SC_{t-1}$ previously computed, and so is $SC_t$. The defined equations are as follows:

$$SR_t = \alpha_1 M SC_{t-1} M^{\mathrm{T}} \cdot NR + (1-\alpha_1) SR_0, nr_{i,j} = \frac{1}{|m_i| \cdot |m_j|} \quad (2)$$

$$SC_t = \alpha_2 M^{\mathrm{T}} SR_{t-1} M \cdot NC + (1-\alpha_2) SC_0, nc_{i,j} = \frac{1}{|m_i| \cdot |m_j|} \quad (3)$$

The $SC_0$ and $SR_0$ represents the initial similarity matrices computed between the embedded representations for each pair of the conditional terms and pattern terms, respectively. Besides, the matrix $NR$ and $NC$ are used for normalization. The parameters $\alpha_1$ and $\alpha_2$ are used to adjust the weight to sum up the derived similarity and the initial similarity, which are set to be 0.1 in our implementations.

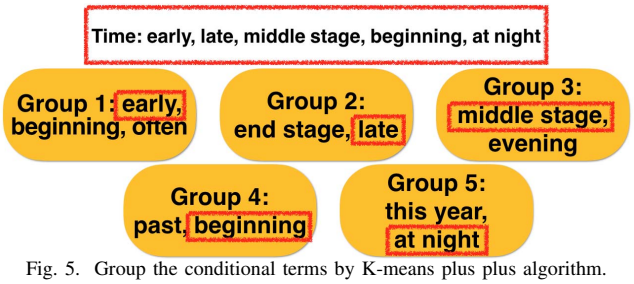



Fig. 5. Group the conditional terms by K-means plus plus algorithm.

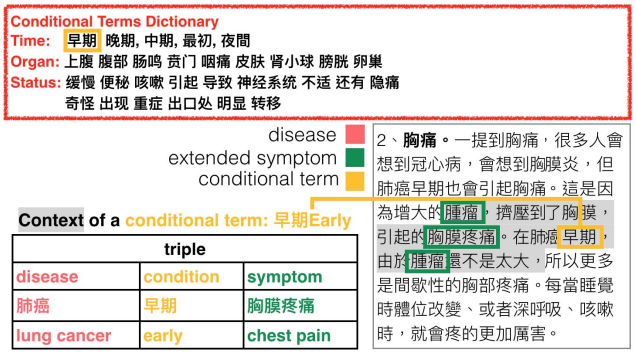



Fig. 6. Extract the conditional relationship triples according to the discovered conditional terms.

According to the Condition-Condition similarity matrix obtained by the co-clustering approach, the conditional terms are then clustered into $k$ groups by performing the K-means Plus Plus algorithm. For example, among the candidate conditional terms with temporal type, the semantically related conditional terms 早期(`early`), 初期(`beginning`), 往往(`often`) are grouped into the same cluster as shown in Fig. 5

Finally, the conditional terms in each cluster with the highest frequency in the disease-related web pages are chosen to be the representative conditional terms. Therefore, in Fig. 5, the representative conditional terms of the 5 clusters are 早期(`early`), 晚期(`late`), 中期(`middle stage`), 最初(`beginning`), and 夜(`at night`), respectively.

For a disease $d$, the discovered temporal, status, and organ representative conditional terms form its conditional terms dictionary. Finally, the conditional terms dictionary as shown in Fig. 6 is used to extract the triple relationships of a disease $d$ having a symptom $s$ with a condition $c$, denoted as `has_symptom`($d$, $c$, $s$). For example, a discovered triple is `has_symptom`(肺癌(`lung cancer`), 早期 (`early`), 胸膜疼痛(`pleural pain`)).

## V. Performance Evaluation

The experiments include two parts: (1) evaluation of the discovered symptoms, and (2) evaluation of the discovered conditional terms.

### A. Data Description

There are 6 diseases selected to be the testing diseases: 肺癌(`lung cancer`), 鼻咽癌(`nasopharyngeal carcinoma`), 糖尿病(`diabetes`), 肝硬化(`cirrhosis`),

TABLE III
THE COLLECTED DATASET FOR EVALUATION.

| | lung cancer | nasopharyngeal carcinoma | diabetes | cirrhosis | colorectal cancer | rectal cancer | in total | avg |
|---|---|---|---|---|---|---|---|---|
| $\#seeds$ | 27 | 7 | 11 | 13 | 12 | 15 | 85 | 14.2 |
| $\#pages_{total}$ | 4272 | 995 | 1861 | 1962 | 1787 | 2256 | 13133 | 2188.8 |
| $\#pages_{avg}$ | 158.2 | 142.1 | 169.2 | 150.9 | 148.9 | 150.4 | 919.8 | 153.3 |

大腸癌(colorectal cancer), and 直腸癌(rectal cancer), which are in the top 10 causes of death in Taiwan. Table III shows the number of seed symptoms, the total number and the average number of the disease-related web pages in search results of the seed symptoms for each disease.

### *B. Evaluation of the discovered symptoms*

For each disease, the discovered symptoms are ranked according to their scoring results of the function $Score(C_i)$ defined in equation (1). This experiment evaluates the Mean Average Precision (MAP) of the discovered symptoms by combining the different methods to compute the important score and related weight used in the equation, respectively.

**Important score**: (1) $RW_C$, (2) $RW_{W2V}$, and (3)$W2V_{avg}$.
**Related weight**: (1) $CP$ and (2) $W2V_{sim}$.

Given the descriptions retrieved from the disease-related web page, which contain both the disease and the discovered symptom. Each discovered symptom was labeled by non-medical experts as 0/1 according to whether the descriptions semantically imply the disease having the symptom. According to the labeled results, the macro average MAPs of the discovered symptoms across diseases were evaluated as shown in Fig. 7 Left. On the other hand, the medical experts are asked to label the discovered symptoms of the diseases score 1 if the disease usually has that symptom, score 0.5 if the disease sometimes has that symptom, and score 0 if the disease seldom has that symptom. According to the labeled scores, a discovered symptom of a disease is judged correct if its score is 1 or 0.5. The macro average MAPs are shown in Fig. 7 Right.

Fig. 7 Left shows that, overall, $RW_{W2V} + CP$ has the best performance, whose macro average MAPs across diseases achieve up to 0.85 and keep stable around 0.8. From MAP@1 to MAP@6, $RW_C + W2V_{sim}$ has the best performance, whose MAP@6 achieves up to 0.9 and 0.85 evaluated by the non-medical and medical experts, respectively. It means that this method can correctly detect the well-known top symptoms of diseases. Moreover, for the three scoring methods of important score, to combine with the $CP$ method for computing the related weight performs better than combined with the $W2V_{sim}$ method from MAP@12 to MAP@100.

According to the results shown in Fig. 7 Right, although the MAP values evaluated by the medical experts are lower than the ones evaluated by the non-experts, their glowing curves have the similar trend. $RW_C + W2V_{sim}$ has the best performance until MAP@35, then catch up by the $W2V_{avg} + CP$. The reason that the MAP evaluated by the experts is lower than the non-expert MAP is discussed as follows. Firstly, some symptoms are too general that the experts don't count it to be the symptoms of the disease. In the case of "There are many patients with colorectal cancer, especially those with colon cancer, found to have a certain degree of anemia in the time of the discovery of the tumor. Anemia can be manifested as dizziness, weakness, cold, dry skin, a headache, insomnia, memory loss, palpitation, shortness of breath, loss of appetite, and gastrointestinal disorders." The non-experts labeled a headache as the symptom of colorectal cancer, while the experts thought a headache is a general symptom that may occur in many diseases. Moreover, some symptoms are caused by the metastasis of the disease. The experts determined that these symptoms are not the symptoms of the disease because the symptoms appear unusually. For example, in the case of "Metastasis causes difficulty in sucking. Hepatic metastases cause hepatomegaly and jaundice or skeletal metastases cause limbs feel sore and so on. After an examination, the symptom is caused by colorectal cancer." That is why he non-experts labeled the hepatomegaly is a symptom of colorectal cancer according to the context but the experts didn't. The proposed method is helpful to construct complete relationships between diseases and symptoms, which provides the noticeable symptoms for further verification in the future.

In Table IV, we compared the MAP@100 for the different diseases. It is interesting that when the disease is lung cancer, using the related weight $CP$ has better performance than $W2V_{sim}$. However, when the disease become diabetes, using the related weight $W2V_{sim}$ performs better than using $CP$. It may because that there are only 11 seed symptoms of diabetes but 27 seed symptoms of lung cancer. More seed symptoms will get more related web pages as the search results. In a sparser dataset, the semantic similarity measure between a pair of candidate symptoms can show their semantic relatedness more effective than computing their conditional probability.

### *C. Conditional Terms Evaluation*

Based on the 100 symptoms discovered by $W2V_{avg} + CP$ scoring method, the corresponding conditional terms are evaluated. The discovered conditional terms are labeled as score 0/1 according to whether the conditional term helps to understand the symptom of a disease more clearly. Accordingly, the precisions of the discovered conditional terms are computed.

In this experiment, we compared the precisions of (1) only using word embedding to compute the similarity of pairs of conditional terms for clustering and (2) the result got by the additional co-clustering step. Furthermore, two baseline methods: Baseline 1 and Baseline 2, are proposed, which choose the top *k* frequent terms from the candidate conditional terms directly. Baseline 1 set *k* equal to the number of conditional terms discovered by the word embedding method and Baseline 2 set k equal to the number of conditional terms

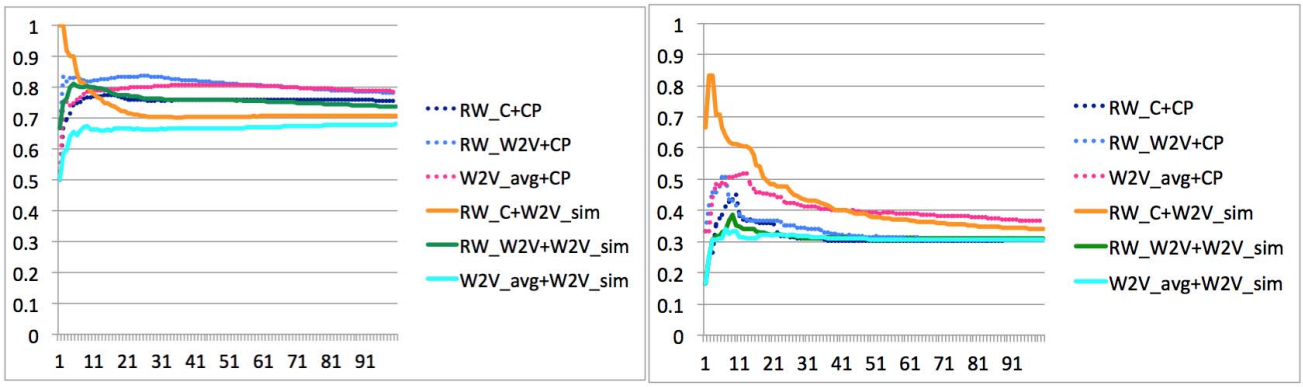



Fig. 7. Left: MAP@1 @100 evaluated by non-medical experts. Right: MAP@1 @100 evaluated by medical experts.

TABLE IV
MAP@100 OF THE DISCOVERED SYMPTOMS BY THE DIFFERENT APPROACHES FOR EACH DISEASE.

| | approach | lung cancer | nasopharyngeal carcinoma | diabetes | cirrhosis | colorectal cancer | rectal cancer | avg |
|---|---|---|---|---|---|---|---|---|
| **Non-expert** | $RW_C$+$CP$ | 0.957 | 0.556 | 0.809 | 0.675 | **0.779** | **0.759** | 0.756 |
| | $RW_{W2V}$+$CP$ | **0.961** | 0.610 | 0.825 | **0.787** | 0.765 | 0.737 | 0.781 |
| | $W2V_{avg}$+$CP$ | **0.961** | **0.675** | 0.832 | 0.759 | 0.744 | 0.744 | **0.786** |
| | $RW_C$+$W2V_{sim}$ | 0.673 | 0.568 | **0.939** | 0.639 | 0.745 | 0.677 | 0.707 |
| | $RW_{W2V}$+$W2V_{sim}$ | 0.678 | 0.647 | 0.882 | 0.739 | 0.762 | 0.700 | 0.735 |
| | $W2V_{avg}$+$W2V_{sim}$ | 0.581 | 0.525 | 0.900 | 0.659 | 0.728 | 0.683 | 0.679 |
| **Expert** | $RW_C$+$CP$ | **0.244** | 0.327 | 0.132 | 0.281 | 0.400 | 0.441 | 0.304 |
| | $RW_{W2V}$+$CP$ | 0.205 | 0.402 | **0.171** | 0.250 | 0.455 | 0.363 | 0.307 |
| | $W2V_{avg}$+$CP$ | 0.202 | **0.493** | 0.141 | 0.374 | **0.533** | **0.454** | **0.366** |
| | $RW_C$+$W2V_{sim}$ | 0.217 | 0.414 | 0.153 | **0.401** | 0.445 | 0.410 | 0.340 |
| | $RW_{W2V}$+$W2V_{sim}$ | 0.150 | 0.391 | 0.128 | 0.351 | 0.410 | 0.424 | 0.309 |
| | $W2V_{avg}$+$W2V_{sim}$ | 0.154 | 0.345 | 0.115 | 0.376 | 0.448 | 0.394 | 0.305 |

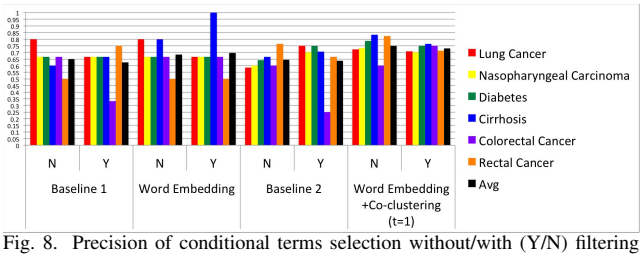



Fig. 8. Precision of conditional terms selection without/with (Y/N) filtering the uncertainty symptoms.

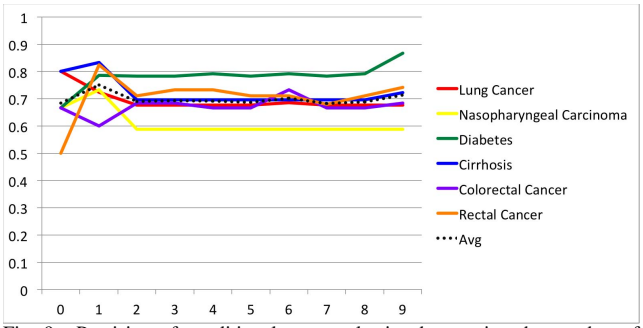



Fig. 9. Precision of conditional terms selection by varying the number of times of co-clustering.

discovered by the word embedding combining co-clustering approach, respectively.

In order to observe the effect of error propagation, we compared the precisions of the conditional terms for all the symptoms discovered in the previous step and the conditional terms for only the symptoms scored by the experts as 1 or 0.5 being remaining. From Fig. 8, it shows that the precisions of the proposed two methods are not affected much no matter the uncertain symptoms are filtered out or not. However, the precisions of the baseline methods do not keep stable for both cases. It also shows that the Word Embedding+Co-clustering approach achieves the highest macro average precision across diseases. About 75% of the discovered conditional terms help further understanding of the relationship between the symptoms and diseases. Therefore, the conditional terms for all the discovered symptoms without filtering are remaining for the following experiments.

Fig. 9 shows the precisions of the discovered conditional terms by varying the number of times of co-clustering from 0 to 9. The results imply that the co-clustering can improve the precisions of the conditional terms for most diseases except `lung cancer` and the `colorectal cancer`. On average, setting the number of times=1 achieves the best performance. According to our observations, the precisions of the result for a disease decreases when performing more times of co-clustering. It is reasonable because when the disease has more candidate conditional terms and context terms, more times of co-clustering will contribute more indirect semantics among the terms. The disease `diabetes` has more focused candidate conditional terms. Accordingly, the precision of the conditional terms for `diabetes` increases up to 0.88 when the number of times of co-clustering is increased to 9. After the 9 times of co-clustering, the disease `diabetes` can find out really helpful conditions such as `skin` and `groin`. For example, "`Diabetic skin pruritus` is

TABLE V
PRECISION OF CONDITIONAL TERMS SELECTION FOR DIFFERENT DISEASES.

| approach | lung cancer | nasopharyngeal carcinoma | diabetes | cirrhosis | colorectal cancer | rectal cancer | avg |
|---|---|---|---|---|---|---|---|
| **Baseline 1** | **0.800** | 0.667 | 0.667 | 0.600 | **0.667** | 0.500 | 0.650 |
| **Word Embedding** | **0.800** | 0.667 | 0.667 | 0.800 | **0.667** | 0.500 | 0.684 |
| **Baseline 2** | 0.586 | 0.600 | 0.643 | 0.667 | 0.600 | 0.765 | 0.644 |
| **Word Embedding+Co-clustering (t=1)** | 0.724 | **0.733** | **0.786** | **0.833** | 0.600 | **0.824** | **0.750** |
| $\#Candidateconditionalterms$ | 2332 | 1134 | 1326 | 1452 | 1638 | 1580 | 1577 |

a common clinical complication of `diabetes`. The clinical manifestation is mainly `pruritus`." Therefore, `diabetes` has symptom `pruritus` on `skin`, which is a conditional term. Moreover, it is helpful to check the symptom `impaired wound healing` on the body part of `groin` when diagnosing the disease `diabetes`.

In Table V, all types of conditional terms are put together. The word embedding combined with co-clustering approach performs the best, the macro precision cross different diseases is up to 0.75.

## VI. CONCLUSION

In this paper, we proposed a system to automatically discover the relationship between diseases and symptoms with conditions from the Internet. The scoring methods are designed to rank the candidate symptoms of a disease extended from the seed symptoms. Embedded representation learning for each candidate conditional term and the co-clustering approach are combined to discover the representative conditional terms. The results of performance evaluation show that the proposed methods can correctly detect the well-known top 6 symptoms of diseases and find the top 100 symptoms with a stable quality 0.78 macro average MAP for the testing diseases. Moreover, the jointly method can effectively discover the conditional terms associated with the symptoms of a disease. In the further, we will extend these strategies to discover the other relationships among the medical concepts with conditional terms.

## REFERENCES


[1] L. Sun, C. Liu, C. Guo, H. Xiong, and Y. Xie, "Data-driven automatic treatment regimen development and recommendation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1865–1874.

[2] M. S. Simpson, E. M. Voorhees, and W. Hersh, "Overview of the trec 2014 clinical decision support track," LISTER HILL NATIONAL CENTER FOR BIOMEDICAL COMMUNICATIONS BETHESDA MD, Tech. Rep., 2014.

[3] R. Feldman, O. Netzer, A. Peretz, and B. Rosenfeld, "Utilizing text mining on online medical forums to predict label change due to adverse drug reactions," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2015, pp. 1779–1788.

[4] K. Lee, A. Qadir, S. A. Hasan, V. Datla, A. Prakash, J. Liu, and O. Farri, "Adverse drug event detection in tweets with semi-supervised convolutional neural networks," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 705–714.

[5] S. Zhang, T. Kang, L. Qiu, W. Zhang, Y. Yu, and N. Elhadad, "Cataloguing treatments discussed and used in online autism communities," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 123–131.

[6] T. R. Goodwin and S. M. Harabagiu, "Medical question answering for clinical decision support," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 297–306.

[7] D. Savenkov and E. Agichtein, "When a knowledge base is not enough: Question answering over knowledge bases with external text data," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 235–244.

[8] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, "Knowledge base completion via search-based question answering," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 515–526.

[9] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute, "Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 5, pp. 507–513, 2010.

[10] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[11] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 101–110.

[12] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5. Atlanta, 2010, p. 3.

[13] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[14] T. Ruan, M. Wang, J. Sun, T. Wang, L. Zeng, Y. Yin, and J. Gao, "An automatic approach for constructing a knowledge base of symptoms in chinese," *Journal of biomedical semantics*, vol. 8, no. 1, p. 33, 2017.

[15] D. Zhang, D. He, S. Zhao, and L. Li, "Query expansion with automatically predicted diagnosis: iris at trec cds track 2016." in *TREC*, 2016.

[16] Z. Wang, Z. Wang, J. Li, and J. Z. Pan, "Building a large scale knowledge base from chinese wiki encyclopedia," in *Joint International Semantic Technology Conference*. Springer, 2011, pp. 80–95.

[17] M. Li, Y. Shi, Z. Wang, and Y. Liu, "Building a large-scale cross-lingual knowledge base from heterogeneous online wikis," in *Natural Language Processing and Chinese Computing*. Springer, 2015, pp. 413–420.

[18] P. Wang, L. Ji, J. Yan, L. Jin, and W.-Y. Ma, "Learning to extract conditional knowledge for question answering using dialogue," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 277–286.

[19] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 653–658.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[22] G. Bisson and F. Hussain, "Chi-sim: A new similarity measure for the co-clustering task," in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 2008, pp. 211–217.

# IExM: Information Extraction System for Movies*

Peng-Yu Chen
National Tsing Hua University
Hsinchu, Taiwan
pengyu@nlplab.cc

Yi-Hui Lee
National Taiwan Normal University
Taipei, Taiwan
amy030619@gmail.com

Yueh-Han Wu
National Tsing Hua University
Hsinchu, Taiwan
dgrey1116@gmail.com

Wei-Yun Ma
Institute of Information Science, Academia Sinica
Taipei, Taiwan
ma@iis.sinica.edu.tw

## ABSTRACT

In this demonstration, we present Information Extraction System for Movies(IExM), which helps extract relation instances from unlabeled movie articles. We have designed a new distant-supervised learning algorithm: Improved Pattern Ranking Algorithm(IPRA) to extract relation instances from unlabeled articles, which iteratively generates new patterns starting from a limited set of seed instances, and extracts new instances using high-ranking pattern in a precise and effective way. IPRA also has a special estimation for the newly generated patterns based on the quality estimation of the instances that generate the patterns and ranks patterns' quality based on various factors.

## Keywords

information extraction; pattern generation; pattern ranking; Wikipedia; E-HowNet; bootstrapping; distant supervision; semi-supervised learning; relation extraction; infobox

## 1. INTRODUCTION

Wikipedia provides infobox to help users gain the information they want conveniently, however, there are still a lot of wiki pages with incomplete infobox. Since manually constructing the infobox is too expensive, we develop a system to automatically extract the structured information from unstructured text data. Combining wiki pages and web resources, we present Information Extraction System for Movies(IExM), which helps extract relation instances from unlabeled movie articles. To complete the system, we need to use some technologies discussed below.

Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine readable documents. To overcome the problem of the lack of training data, Mintz *et al.* [1] proposed Distant-supervised learning algorithm (DSLA) to generate training data from Freebase. Another strategy is through bootstrap learning to extract more patterns from seeds or instances in an iterative fashion. However bootstrapping often suffer from semantic drift problem [2]. To address this issue, two approaches are common in use. NELL [3, 6, 7] proposes the coupled training method by building a large amount of coupled relations and setting the mutually exclusive constraints between these coupled relations. Sun and Grishman [4] designed a pattern ranking algorithm with pattern clustering strategy to prevent semantic drift. However, their method does not update patterns' quality and also fail to consider the quality of the instances that generate these patterns.

To address the above considerations, we propose Improved Pattern Ranking Algorithm (IPRA) for IE tasks, which estimates patterns' quality according to various factors, including the patterns' occurrence and coverage of application, and the quality estimation of the instances which are actually extracted by these patterns. The experimental results show that as more patterns are generated and ranked, the coverage and precision of extracted instances can be gradually improved and then achieve a high performance in the end.

The related work are introduced in Section 2. In Section 3 and 4, the details of our system and IPRA model are introduced. The performance evaluation on the proposed methods and related works is reported in Section 5.

## 2. RELATED WORK

At least four learning paradigms of information extraction have been presented for the task of extracting relation from text. First paradigm is to manually design patterns for a rule-based approach, which is born with the defects that it takes much human efforts and lacks flexibility. The second paradigm is through a supervised learning procedure: to build a large-scale, machine learning classifier to judge if a given entity pair has a certain relation. Since it requires a large amount of labeled data, a lot of manual efforts are also needed.

Another common paradigm is through bootstrapping method for semi-supervised learning [2, 5]: to begin with a limited number of labeled instances in context and much more unlabeled documents in a specific domain, and extract patterns as extractors. The extracted instances are used with a large corpus to generate a new set of patterns. The generated patterns are then used to extract more instances. Each time the process involving a stage of "instances gen-

*The work described in this paper was done as part of (and partially supported by) 2016 Summer Internship Program at the Academia Sinica, Institute of Information Science, Taiwan.


*WWW 2017 Companion,* April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054729

erate patterns" and a stage of "patterns extract instances" is called an iteration. Brin *et al.* [2] present a technique which exploits the duality between sets of patterns and relations to grow the target relation starting from a small sample, and test the extract relation (author,title) pairs from the World Wide Web. However, it often causes "semantic drift" problem after many iterations [2, 5]. e.g., For extracting the 'country' from the movie articles, 'the United States' is the target instance of the pattern 'film in', but the pattern generated from 'the United States' could be 'live in'. New instances generated from the pattern 'live in' may be 'apartment building'. In this situation, we get the instance 'apartment building', but it's not an instance of 'country'. The semantic meaning of the target instance deviates.

For solving the semantic drift problem efficiently, there are two methods which are commonly used. One is coupled training, the other is pattern ranking. NELL [3, 6, 7] makes use of coupled training method to build a large number of coupled relations. With an initial ontology defining categories and about a dozen labeled training examples for each category and relation, NELL extracts candidate instances by patterns and evaluates the quality of the candidate instances by the number of promoted patterns that they co-occur with [3]. They also set mutual exclusive constraints between these coupled relations. NELL has been learning to read the web 24 hours/day since January 2010, and so far has acquired a knowledge base with over 80 million confidence-weighted beliefs (e.g., servedWith(tea, biscuits)). On the other hand, Sun and Grishman [4] design the pattern ranking algorithm. A pattern ranking algorithm with pattern clustering strategy is presented to prevent semantic drift. While pattern clustering strategy does bring benefits, their framework only estimates patterns' quality based on the instances (and their clusters) that these patterns can match, and accept a certain number of top ranked patterns. Their method does not update patterns'qualities based on the instances that the top ranked patterns actually generated and also fail to consider the quality of the instances that generate the patterns.

Another similar paradigm is Distant-supervised learning algorithm (DSLA). It is similar with semi-supervision, and the only difference is the seeds/instances are usually certain target objects, such as attributes, instead of labeled instances in context.

For the pattern design, context pattern [2, 5] and syntactic pattern are common in use. Our work investigates syntactic patterns and mixed context patterns, combining three different semantic units in the pattern design: word, part-of-speech tag and word sense.

## 3. SYSTEM DESCRIPTION

In this section, we will describe how our system, IExM, works based on IPRA model. The system is presented in the form of a web application[1], where you can input a movie title and the attribute you wish to know. Once you click the 'search' button, the system will collect articles from Wikipedia and other websites related to the movie, extracting the target attribute and list the result. The system will also highlight the target attribute and the pattern in the result. Figure 1 shows the system screenshot, which we will demonstrate at the conference. You can also find the screencast in the same website.

The system architecture (as illustrated in Fig. 2) consists of three main components, the pretrained model using Wikipedia data with IPRA(we will describe this model in detail in next section), the Wikipedia database, and a user interface. Once a user types the movie name with an attribute which he wants to search, our system first collects some related articles from the web with the movie

[1]http://learn.iis.sinica.edu.tw/IExM

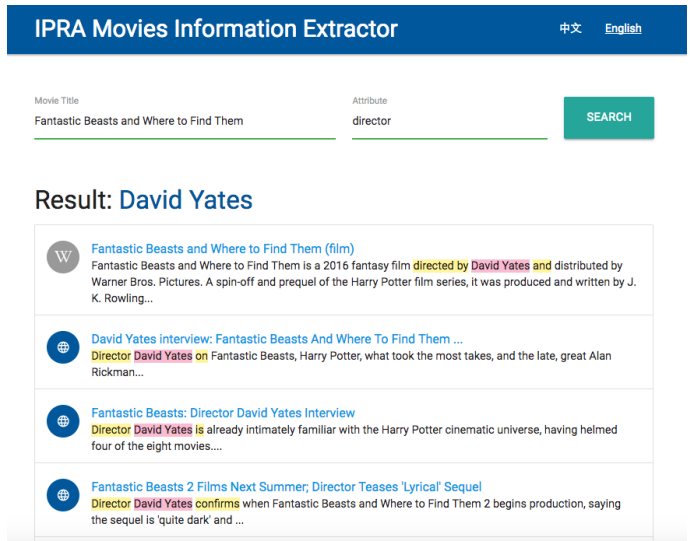



**Figure 1: The screenshot of the system**

name as keyword, then for every article, we try to find a pattern in our model that can match this article. Based on all the matched articles and the information in Wikipedia database, the system chooses an appropriate answer and shows the result to user.

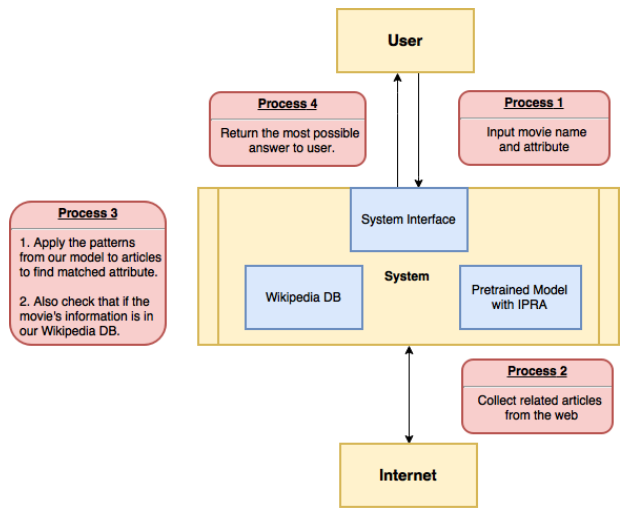



**Figure 2: System architecture**

We are still extending the data domain of our system. In the future, our system will have a variety of knowledge in different domains in addition to movie information, and this would also be useful on many NLP tasks.

## 4. MODEL

We developed an information extraction model with an Improved Pattern Ranking Algorithm, named IPRA. IPRA can extract attributes from an article of a specific theme. The attribute values extracted are called **instances**, and the initial manually chosen instances are **seeds**. To simplify the problem, we focus on Chinese Wikipedia articles and choose movies and TV series as our domain categories.

## 4.1 Pattern Design

We have two main types of pattern: the context pattern and the syntactic pattern. A context pattern consists of the context information of the target attribute, and a syntactic pattern focuses on the sentence structure in which the target attribute occurs.

### *4.1.1 Context Pattern*

Suppose we have a sentence:

《斷背山》由台灣導演李安執導
**English:** Brokeback Mountain is directed by Ang Lee, a Taiwanese director.

After Chinese word segmentation and part-of-speech tagging, we get the result shown in Figure 3.

| POS | parenthesis | Nc | parenthesis | P | Nc | Na | Nb | VC |
|---|---|---|---|---|---|---|---|---|
| word | 《 | 斷背山 (Brokeback Mountain) | 》 | 由 (by) | 台灣 (Taiwan) | 導演 (director) | 李安 (Ang Lee) | 執導 (directed) |

**Figure 3: Word segmentation and POS tagging**

Our target attribute value, the director of the movie, is 李安(Ang Lee). Now we need to look at the context of the word and transform that into a pattern. Assuming the window size of the context is 1, and the target attribute is denoted as (.+?) in regular expression, we have four types of context pattern described below:

1. **Word**
   Only looking at the left adjacent word and the right adjacent word of the target.
   **pattern:** <導演>(.+?)<執導>
   (<director>(.+?)<directed>)

2. **POS**
   A POS(part-of-speech tagging) pattern can extract more attributes than a word pattern can, although this may also decrease the precision.
   **pattern:** <Na>(.+?)<VC>

3. **E-HowNet word sense**
   The term 'word sense' means a general representation of a word. The definition of sense we adopt is based on E-HowNet(Extended-HowNet)[2].
   **pattern:** <human|人.1>(.+?)<undertake|擔任.1>

4. **Mixed**
   Combining the three types above, we create a mixed pattern type. Assuming the context window size is 2, we have 4 positions for 3 kinds of pattern type. So the number of different patterns is $3^4 = 81$

   - word word (.+?) word wod
   - word word (.+?) word pos
   - ...
   - sense sense (.+?) sense pos
   - sense sense (.+?) sense sense

[2]http://ehownet.iis.sinica.edu.tw/index.php

### *4.1.2 Syntactic Pattern*

Given the sentence '《斷背山》由台灣導演李安執導', we can obtain a syntactic tree structure by using CKIP Chinese Parser. Leaf nodes being the segmented words, each internal node has two values, the semantic role and the part-of-speech tagging of the subtree. Leveraging on this tool, we create two kinds of syntactic patterns.

1. **Parse tree path**
   Taking the semantic roles along the path from root node to the node of our target attribute, we create a pattern that can carry some syntactic information of the sentence.

2. **Parse tree path with head word**
   Only looking at the tree path may be too ambiguous. A path consisting of some semantic roles might have nothing to do with the attribute itself. For this reason, we add another factor, head word, into the pattern, In a syntactic tree, head word can usually capture the key intention of the sentence.

## 4.2 Improved Pattern Ranking Algorithm

With a sample of seeds, IPRA iteratively generates patterns and extracts attributes from chosen Wikipedia articles. Figure 4 shows the framework of IPRA. A single iteration of the process consists of four stages, which is described below.

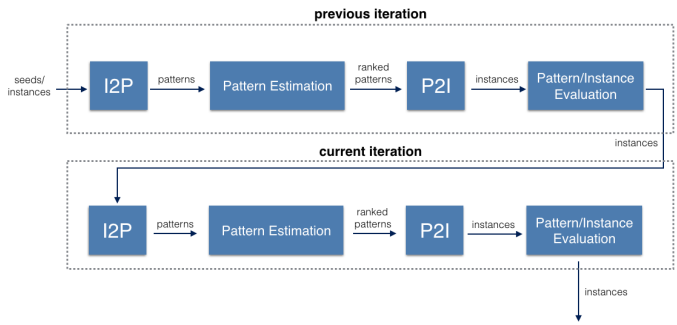



**Figure 4: IPRA framework.**

### *4.2.1 Instance-To-Pattern Stage(I2P)*

With given attribute value in an article, we find the sentences where the attribute occurs, and transform the way the attribute is mentioned into our pattern format.

### *4.2.2 Pattern-To-Instance Stage(P2I)*

The patterns generated through instance-to-pattern stage can then be used to extract new instances. Before starting the extraction process, we use our pattern ranking algorithm to rank the new generated patterns and all the previously generated patterns. The pattern with the highest rank would be used to extract attributes first, and the patterns with lower rank would only be used on the articles whose attribute values are not yet extracted. To observe the performance of each iteration, before going back to the instance-to-pattern stage, we evaluate all the instances using the infobox of each article and calculate the precision and coverage, shown in Figure 5.

### *4.2.3 Pattern/Instance Evaluation*

In pattern-to-instance stage, how should we decide which pattern to apply first if there were several patterns to choose from? We propose to estimate patterns' quality according to various factors,

including the patterns' occurrence, the coverage and the quality estimation of the instances which are actually extracted by these patterns. Based on these assumptions, we developed a new pattern ranking algorithm to score a pattern before and after it is used in pattern-to-instance stage. In the rest of this subsection, we will introduce how to estimate the quality of pattern after P2I and introduce the pattern estimation before P2I on 4.2.4.

First, we define how to measure the quality of an instance. We call it **precision**. The precision of seed instances are initialized to 1, while the precision of other instances are initialized to 0. Suppose the instance $I_i$ is obtained from $k$ source patterns. At the end of each iteration, we recalculate the precision of each instance using the equation below:

$$Prec(I_i) = \frac{\sum_{j=1}^{k} Conf(P_j)}{k} patternbeforeandafter \quad (1)$$

$Conf(P_j)$ is the **confidence** of pattern $P_j$, which is described below. To calculate the score of a pattern, we consider three factors:

1. **Term frequency(TF)**: The number of times the pattern being applied in the corpus.

2. **Document frequency(DF)**: The number of documents in which the pattern is applied.

3. **Confidence(Conf)**: A value indicating whether the pattern is generated from a good instance. Suppose the confidence of a pattern $P_i$ is denoted as $Conf(P_i)$, and the precision of its $k$ source instances $I_j$ is $Prec(I_j)$, the equation is:

$$Conf(P_i) = 1 - \prod_{j=1}^{k}(1 - Prec(I_j)) \quad (2)$$

The equation 2 shows that the confidence of a pattern would remain to be 1 if any of its source instances comes from the seed instances. We normalize the value of each factor to [0, 1] and calculate the score by taking the average of the three values.

$$PatternScore(P_i) = \frac{TF(P_i) + DF(P_i) + Conf(P_i)}{3} \quad (3)$$

We keep track of instance precision and pattern score by maintaining an instance precision table and a pattern ranking table, updating the values in the end of each iteration.

### *4.2.4 Pattern Estimation*

When a new pattern is generated from an instance, we can calculate its confidence, but how can we know the TF and DF before it is applied to extract attributes? We can not calculate the real score. Therefore, we present a special estimation for the newly generated patterns based on the quality estimation of the instances that generate the patterns, and these quality estimation of the instances are related to the source patterns that generate these instances. We define the $InstanceScore$ of an instance to be the weighted average of the scores of all of its source patterns. The weight is the reciprocal of the rank number of the pattern.

$$InstanceScore(I_i) = \frac{\sum_{j=1}^{k}(PatternScore(P_j) \times \frac{1}{rank(P_j)})}{\sum_{j=1}^{k} \frac{1}{rank(P_j)}} \quad (4)$$

Then, we can calculate the $EstimatedPatternScore$ of the newly generated pattern by taking the average of its source instances' $InstanceScore$.

$$EstimatedPatternScore(P_i) = \frac{\sum_{j=1}^{k} InstanceScore(I_j)}{k} \quad (5)$$

The estimated score of the pattern is overwritten by the real score once the pattern finishes pattern-to-instance stage.

# 5. EXPERIMENT

## 5.1 Data and Preprocessing

We dump all Chinese articles from Wikipedia as our experimental resource which include both unstructured text and infobox. An infobox is a fixed-format table on Wikipedia page designed to consistently present a summary of some unifying aspect that the articles share. We collect articles list from Wikipedia's categories called movie and TV series and all its subcategories, which contain 4694 movie articles and 5817 TV series articles, as our domain, making use of the information from the infobox such as director, country, and screenwriter to produce initiative seeds and also take the infobox as the golden answer to evaluate the result.

In order to generate part-of-speech and parse tree for the syntactic and context pattern, we use CKIP Chinese Word Segmentation System[3] and CKIP Chinese Parser[4]. These tools have high accuracy in Chinese environment compared to others. Besides, we also use the Extended-HowNet(E-HowNet) to expand our context pattern. E-HowNet is the lexical semantic representation model for natural language understanding. Changing the context pattern from word layer to sense layer enhances the ability to match more instances.

## 5.2 Result

Each article could contain varying attributes in the infobox. For extracting the specific attribute to evaluate our method, we select three attributes: "director", "country" and "screenwriter". We pick up 4105, 3895, 710 articles from 4694 movie articles and 5817 TV series articles which have attribute "director", "country" and "screenwriter" in the infobox respectively. For held-out evaluation experiments, we randomly pick up 20 articles as our seeds to see if our method could obtain the specific attribute from the rest articles.Our method gets better F1-score when we use pos-based context patterns and set the window size to 2. And word-based context patterns gets better precision. The result is shown in Table 1 and 2.

**Table 1: Result of Different Attributes**

| | Precision | Recall | F1-Score |
|---|---|---|---|
| Director | 86.1% | 63.8% | 73.3% |
| Country | 80.1% | 69.4% | 74.4% |
| Screenwriter | 99.0% | 55.6% | 71.2% |

### *5.2.1 Pattern Ranking Algorithm*

We choose the best pattern type: **pos pos target pos pos** to compare the performance of two baselines and our IPRA algorithm. The two baselines are **Voting**: collecting the instances extracted by new patterns in one article, and then decide the instance

[3]http://ckipsvr.iis.sinica.edu.tw/
[4]http://parser.iis.sinica.edu.tw/

**Table 2: word/pos/sense/mixed(top4) Context Patterns**

| Pattern Type | Precision | Recall | F1-Score |
|---|---|---|---|
| word word target word word | **90.2%** | 55.3% | 68.6% |
| pos pos target pos pos | 86.1% | **63.8%** | **73.3%** |
| sense sense target sense sense | 89.7% | 56.0% | 68.9% |
| pos pos target pos word | 85.7% | 63.2% | 72.7% |
| pos pos target pos sense | 85.7% | 63.4% | 72.7% |
| word pos target pos pos | 87.8% | 61.9% | 72.6% |
| word pos target pos word | 88.0% | 61.6% | 72.5% |

with highest votes, and **Confidence-Based Pattern Ranking Algorithm (PRA)**: considering only the confidence of pattern to estimate the patterns' quality, that is, using only equation 1 and equation 2. Figure 5 obviously shows that our approach IPRA(f1-score: 0.733) performs better than voting method(f1-score:0.670) and confidence-based PRA(f1-score: 0.680), which shows the reasonable assumption that the dynamic evaluation for pattern enhances the performance.

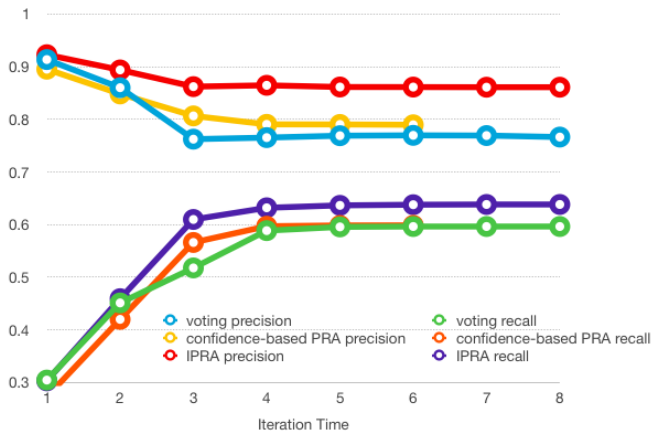



**Figure 5: Algorithms' performance comparison**

### 5.2.2 *Missing Information in the Infobox*

We manually evaluate about 600 articles which do not have the attribute "director" in the infobox. By human evaluation, the attribute "director" is mentioned in the 179 articles. With our method, we find out the director from the 101 articles among the 179. Although there are 78 articles missed or have wrong answer, we can still get a precision of 77% and recall of 56% which shows in Table 3. The experiment prove that our method has the potential to extract information from the context correctly, and it can be applied to a variety of tasks including expanding Wikipedia and other corpus.

**Table 3: 589 articles which miss 'director' attribute**

| | Found | Not found |
|---|---|---|
| Director appears in context | 101 | 78 |
| Director doesn't appear in context | 31 | 379 |
| Precision: 77% , Recall: 56% , F1-Score: 65% | | |

## 6. REFERENCES


[1] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. *Distant supervision for relation extraction without labeled data*. In AFNLP.

[2] S. Brin. *Extracting patterns and relations from the world wide web*. In WebDB Workshop at 6th Intl. Conf. on Extending Database Technology, 1998.

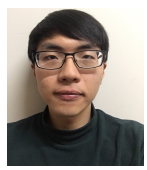
[3] A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr. and T.M. Mitchell. *Coupled Semi-Supervised Learning for Information Extraction*. In WSDM, 2010.

[4] Ang Sun and Ralph Grishman. 2010. *Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters*. In COLING.

[5] Ellen Riloff and Rosie Jones. *Learning dictionaries for information extraction by multi-level bootstrapping*. In Proc. of AAAI, 1999.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr. and T.M. Mitchell. *Toward an Architecture for Never-Ending Language Learning*. In Proceedings of the Conference on Artificial Intelligence (AAAI), 2010.

[7] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling. *Never-Ending Learning*. In Proceedings of the Conference on Artificial Intelligence (AAAI), 2015.

**Peng-Yu Chen** received the B.S. degree of Computer Science from National Tsing Hua University, Hsin-Chu, Taiwan in 2014. He is currently working towards the Graduate degrees of Computer Science, National Tsing Hua University. His research interests include natural language processing, data mining, and web development.

**Yi-Hui Lee** received the B.S. degree of Computer Science from National Taiwan Normal University, Taipei, Taiwan, in 2015. She is currently working towards the Graduate degrees of Computer Science at National Taiwan Normal University. Her main research interests include data mining, information retrieval, machine learning, natural language processing.

**Yueh-Han Wu** is currently pursuing B.S. in Computer Science at National Tsing Hua University, Taiwan. He spends lots of time doing web development and natural language processing.

**Wei-Yun Ma** received his M.S. degree and Ph.D. degree from Columbia University in 2008 and 2014, respectively. Currently, he is an assistant research fellow of the Institute of Information Science, Academia Sinica, focusing on sematic analysis of social media and machine reading.

His research interests include natural language processing, natural language understanding, machine learning, deep learning, machine translation and knowledge representation.