IPA

2010 年度未踏 IT 人材発掘・育成事業　採択案件評価書

1. 担当 PM

平本　健二 PM（経済産業省　CIO 補佐官）

2. 採択者氏名

チーフクリエータ：NGUYEN　TUAN　DUC（東京大学大学院　情報理工学系研究科　創造情報学専攻）

コクリエータ：BOLLEGALA DANUSHKA TARUPATHI（東京大学　情報理工学系研究科　電子情報学専攻　助教）

3. 委託金支払額

2,880,000 円

4. テーマ名

言語横断型の潜在関係検索エンジンの開発

5. 関連Webサイト

http://www.milresh.com/

6. テーマ概要

WWW 空間における情報爆発に伴い、単純なキーワードを含む Web ページ検索だけではなく、エンティティ間の関係に着目した関係検索も有効な場面が考えられるようになってきた。その中で最も有効と考えられるのが潜在関係検索である。

本プロジェクトでは、{(東京，日本)，(?，France)}というようなクエリに対して、はてな（?）の位置に適切な答え「Paris」を出せるように、言語横断型の潜在関係検索エンジンを開発した。本プロジェクトの特徴は WWW に書かれている文書をコーパスとし、言語に跨った文を根拠とした検索を可能にすることである。これにより、異なる言語間のエンティティ検索ができ、かつ、類似意味を持つ文を異なる言語間で検索ができるようになった。

7. 採択理由

提案者が指摘するように多言語であっても文書解析せず関係性に着目するという点は、従来の検索サービスとも正面からぶつかるものではなく、面白いアプローチである。実際に、「〇〇における××」というのは日常的にも良く遭遇する疑問であることから、実用性もあると考え採択した。行政機関においても、「建築業界における業者登録制度は、情報処理業界における〇〇制度」や「日本の雇用統計は、フランスの＊＊」であるなどの展開が考えられる。行政機関は定型化された多くの情報が蓄積されていることもあり有効な利用が期待される。

また今後の、ソフトウェアの技術開発においてグローバル展開は重要であり、最初からグローバルに対応できるところも評価ができた。

プレゼンでは、クリエータ、コクリエータのまじめさが伝わってきており、その点も評価できた。

8. 開発目標

本プロジェクトの目標は高速・高精度の単一言語、言語横断型の潜在関係検索を実現することである。潜在関係検索の例を図 1 に示す。

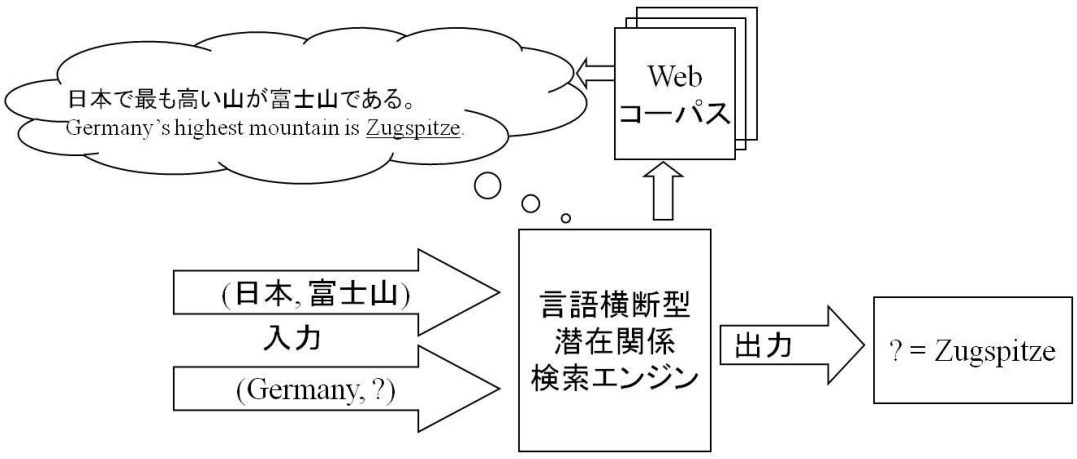

図 1. 言語横断型の潜在関係検索の例

図 1 の例では、クエリ{(日本, 富士山), (Germany, ?)}が入力された際に、日本語のテキストにおける「日本」、「富士山」に関連する文章から「最も高い山」などの関係を抽出し、英語のテキストから「the highest mountain in」などの関係を持つ(Germany, ?)のペアを検索する。

9. 進捗概要

(1) アーキテクチャ設計、データベーススキーマ設計

本検索エンジンのアーキテクチャは図 2 に示すように、Database-Centric とコンポーネントベースアーキテクチャにした。

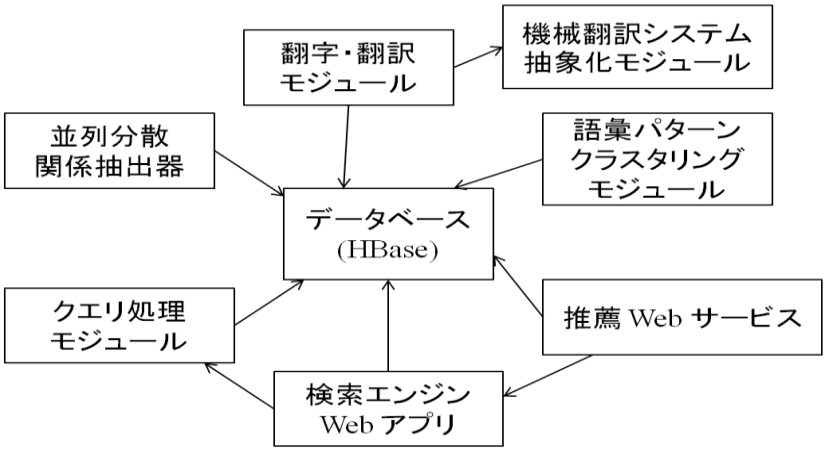

図 2. 開発した検索エンジンのアーキテクチャ
（矢印の方向は利用コンポーネントから利用されるコンポーネントまでを示す）

これは検索エンジンの Index が最も重要な要素であり、各コンポーネントは常に変更が要求されると認識したからである。Database は高速に大量データを管理、処理できるために、key-value store 型の HBase を用いた。他のコンポーネントは独立のパッケージで構成され、独立のプロセスで実行される。各プロセスのプロセス間通信はデータベースを利用して行う。

コンポーネントの中の実装は自由に変更でき、他のコンポーネントには影響を与えずに規定された機能を提供できる。また、データベーススキーマでは、関心毎に HBase のテーブルに分けている。

(2) 並列分散関係抽出器・索引作成モジュールの実装

潜在関係検索を高速に行うために、潜在関係検索の専用の Index を作成する必要がある。その最初の手順として、テキストコーパスから文を取得し、取得された文に対して、固有表現抽出器（MeCab と Stanford Named Entity Recognizer）を適用した。固有表現抽出器により、エンティティを認識し、エンティティペアを抽出した。また、高速にコーパスを処理するために、Hadoop MapReduce で並列に文書を処理することにした。Map の段階では、文からエンティティペアの抽出、ペアの関係を特徴付ける語彙パターン抽出を行い、エンティティペア、語彙パターンを emit する。

Reduce 段階では、Map で emit されたエンティティペアと語彙パターンを HBase の各テーブルに保存する。

例えば、文「東京は本州にある都市で、日本の首都である」からは、3 つのエンティティ「東京」、「本州」、「日本」が抽出され、3 つのエンティティペア「(東京，本州)」、「(本州，日本)」、「(東京，日本)」が抽出される。上記で抽出されたエンティティペアにおける 2 つのエンティティの関係を表現するために、そのエンティティペアの周辺の文脈の語彙パターンを使った。語彙パターンはエンティティペアの周辺の文脈の n-grams である。例えば、表 1 に示すように、文「大都市の東京は日本の首都である。」でのエンティティ「東京」と「日本」との関係は語彙パターン「都市 X は Y の首都」や「X は Y の首都」、「X * Y の首都」などで表現される。ここで、ワイルドカード記号「*」は 1 つ以上の文字を表し、変数 X と Y は元々のエンティティを表す。元々のエンティティを変数 X と Y で置き換えたのは、語彙パターンを比較できるように、各エンティティペアに依存しないようにするためである。

表 1. 語彙パターン抽出プロセスの例

形態素解析	大|都市|の|□東京|は|日本|の|首都|である。
エンティティ認識	大 都市 の 東京 は 日本 の 首都 である。
変数置き換え	大 都市 □□X は Y の 首都 である。
2 文字周辺□脈抽出	都市□□は Y の首都
語彙パターン生成	都市の X は Y の首都, X は Y の首都，の X は Y の首都, X *□Y の首都，…

また、HBase に保存された後、エンティティペアがどの語彙パターンと何回出現したのかを表す特徴量を表すベクトルができる。

(3) 翻字・翻訳モジュールの実装

言語横断型の検索を行うために、異なる言語で表現される類似の意味関係を認識する必要がある。そこで、本プロジェクトでは翻字・翻訳モジュールを使い、言語横断的に類似関係を認識した。

ある言語から他の言語に表記を系統的に変換することは翻字（transliteration）プロセスである。本プロジェクトでは、「パリ」と「Paris」が同じであると認識するために、この翻字プロセスを実行した。翻字した後、例えば、ペア(ビルキルカラ，マルタ)と(Birkirkara, Malta)が同じと認識され、語彙パターンが合併される。

また、語彙パターンの翻訳も行った。関係抽出プロセスで、語彙パターンの中のスロット X と Y のタグ（固有名詞の種類：組織名、人名、地名など）が分かるので、X と Y に適切な固有名詞を入れて、翻訳のためのフレーズを作り、翻訳モジュールの入力として使った。翻字・翻訳モジュールでは、機械翻訳システムが簡単に変更できるように、

Factory Method のデザインパターンを使った。現在の実装では、グーグル翻訳 API と Moses 翻訳ソフトが利用できる。

(4) クラスタリングモジュールの実装・インクリメンタル化・高速化

翻字や翻訳のプロセスを実現したとしても、類似するエンティティペアの間で共通する語彙パターンはまだ少ない可能性がある。これは、1 つの関係に複数の言い換え表現があるからである。例えば、「X が Y の最大都市である」や「X が Y の最も大きい都市である」は実際に同じ関係を記述しているが、言い回し的には異なる表現である。そこで、この 2 つの表現が意味的に類似することを認識する必要がある。このために、本プロジェクトのコクリエータが提案した「逐次語彙パターンクラスタリング」手法を使い、類似する語彙パターンをクラスタリングする。語彙パターンの類似度はパターンと一緒に出現したエンティティペアの頻度ベクトルのコサインとして定義する。逐次語彙パターンクラスタリングを実行した後、類似したパターンは 1 個のクラスタにまとめられる。

更に、対訳語彙パターンを持つパターンについては、対訳語彙パターンについて、ソフトクラスタリングを行った。これにより、類似意味の多数の異なる言語の語彙パターンが 1 つのクラスタにまとめられ、言語横断型の検索精度を向上できた。

また、本プロジェクトでは、検索エンジンが動作しながら、Indexing を行うので、新しい語彙パターンが生成される。そこで、クラスタリングもインクリメンタル的に行う必要がある。そこで、語彙パターンクラスタリングのモジュールを完全に抽出器やクエリ処理モジュールから分離し、独立のプロセスとして実行することにより、上記の問題を解決した。

(5) クエリ処理モジュールの実装、検索の精度の最適化

ソースペアの語彙パターンと対訳パターン情報や語彙クラスタの情報を使い、類似するパターンを持つペアを探し、結果を出力するモジュールを実装した。即ち、クエリ{(A, B), (C, ?)}が入力された際に、まず、Index から入力されたソースペア(A, B)を検索し、このペアの語彙パターンを取得する。次に、キーエンティティ(C)を含むペアで、(A, B)と 1 つ以上の語彙パターンを共有するかまたは(A, B)の語彙パターンと同じクラスタにある語彙パターンを含むペアを検索する。語彙パターンからその語彙パターンと一緒に出現したエンティティペアへの転置 Index を用いて、この操作は高速に行える。これらのペアを候補ペアとした。

検索結果のランキングをするために、エンティティペア間の関係類似度を計算する機能をこのモジュールで実現した。

(6) キーワードで関係絞り込みの機能の実装

ソースペアにおける 2 つのエンティティの間には、複数の意味関係を持つ場合がある。そこで、ユーザがある特定の関係を検索したい時に、関係を明示に検索エンジンに指定できるように、キーワードによる関係絞り込みの機能を実装した。この絞り込みキーワードは特定の関係に興味を持つときだけ入力するオプションの入力である。

関係絞り込みのキーワードが入力されたときに、このキーワードを含む語彙パターンだけを検索対象にすることで関係の絞り込み機能を実現できた。即ち、候補を取得する際に、共通の語彙パターンにこのキーワードを含むという制約を入れることで、絞り込み機能を実現した。

(7) 検索エンティティ推薦機能の実装

ユーザが検索したいエンティティを入力する際に、エンティティの部分マッチングを行い、エンティティが完全に入力されなくても、エンティティの表記を推薦する機能を実装した。

(8) クローラの調整、クローリングとコーパス解析、Index 化

検索エンジンの Index を作るために、1.6GB のウェブページコーパスをクローリングした。また、今後さらにデータを Index に導入できるようにするために、ClueWeb09 コーパスを購入した。これにより、大量の Web ページのコーパスを手に入れることができたので、インクリメンタル的にこのコーパスを解析し、Index 化を行っている。

(9) テスト・性能最適化

クラスタリングの類似度閾値などに、適切なパラメータ値を特定することができた。

また、大量クエリを処理できるように、検索エンジンのウェブアプリを現在 2 つのマシンで並列に実行している。外側では 1 つの IP しか見えないが、中では、Linux Virtual Server（LVS）を使い、クエリを分散している。これによりマシンの数が多ければ多いほど処理できるクエリの数が増える。

10. プロジェクト評価

アーキテクチャに従い解決を行ったため、開発が堅実であり、一つ一つの要素について丁寧に開発を行っている。また、様々な言い回しに対応できるように、並列分散関係抽出器と索引作成モジュールを使ったり、逐次語彙パターンクラスタリングを使うなど工夫をしている。開発自体は完成しているが、ユーザインタフェースの改善や大量データの処理など、残った課題もある。しかし、小規模の追加開発で実用的に使える域まで達しており、プロジェクトとしては成功したと言える。今後の開発と展

開に期待したい。

11. 今後の課題

プロジェクトの期間内でソフトウェア開発を完了したが、大量のデータを使い、検索エンジンの索引を作成するまでは時間が足りなく、まだ実現途中である。現在、検索エンジンの索引を徐々に増やしており、近い将来には実用レベルのデータ量が解析されると期待している。まず、政府関係のデータに集中して索引を作成する予定であり、出来上がったときに、経済産業省のオープンガバメントラボ（http://openlabs.go.jp/）で公開できるよう進める予定である。