日本語テキストの合成演算

織 学　　佐藤 理史

京都大学大学院 情報学研究科 知能情報学専攻

1. はじめに

テキスト形式で記述された知識を知識源として利用する自然言語処理応用システムには、応答文を生成するための明示的なテキスト生成機能をもたないシステムがある。たとえば、黒橋らの対話システム [1] では、計算機システムの使い方に関する知識を「Xするためには、Yして下さい」のような形式のテキスト(文)で記述しておき、「Xしたいのですが」のような質問に、この文の後半部である「Yして下さい」で応答する方式が採用されている。また、桜井らの用語説明探索システム [4] では、与えられた用語を説明する文章として、ウェブから切り出した文章をそのまま表示するという方式が取られている。

このようなシステムは、典型的な質問に対して、かなり良い応答を返すことができる。しかしながら、基となるテキストを加工する能力をほとんど持たないため、その能力には限界がある。この限界を越えるためには、質問に対する適切な答を動的に作成できるような機構、すなわち、テキストを自在に加工する能力の実現が必要である。

本研究では、このような能力を実現するための第一歩として、「2つの文から、それを合成した文(または文章)を作成する」というテキスト合成演算を実現する方法について検討する。これは、いわば、テキストの「加算」に相当する演算であり、2つの文に別々に書かれている情報を統合する手段を与えるものである。

2. テキスト合成演算の定義

ここで考える演算は、次のようなテキストの「加算」を行うものである。

X =「JUMANは形態素解析システムです」
Y =「形態素解析システムは、文を単語に分割する」
$Z = X + Y$
　=「JUMANは、文を単語に分割する形態素解析システムです」

本研究では、この演算を次式のように *add-on* という5引数の関数でモデル化する。

$$Z = add\text{-}on(X, R, Y, C, G) \quad (1)$$

ここで、X と Y は合成すべき2つのテキストであり、R はその間の関係を表す。関係 R を明示的に導入するのは、任意のテキスト X と Y に対して合成演算が実行可能なわけではなく、X と Y にある種の関係 R が設定できる場合のみ、合成演算が可能となるからである。すなわち、テキストの「加算」を、「テキスト X に対して、R という関係でテキスト Y を追加したテキスト Z を作成する」と捉える。

C は、生成されるテキスト Z が満たすべき明示的な制約(たとえば、「1文」や「50文字以内」)を表す。また、G は、Z が自然言語テキストとなるための指針を表す。この指針を完全な形で列挙することはできないが、おおよそ、1) 文章を構成するそれぞれの文が構文的・意味的に正しい文となっていること、2) 文章全体に結束性があり、代名詞・省略の使用などの点で自然な文章となっていること、などが含まれると考える。

式(1)の出力 Z は、以下のことを満たすことを条件とする。

(a)　自然言語テキストとなるための指針 G を満たす
(b)　関係 R が読みとれる
(c)　X の伝える内容が読みとれる
(d)　Y の伝える内容が読みとれる
(e)　外部からの制約 C を満たす

なお、本稿では、「自然言語テキスト」という言葉を「自然言語として意味の通るテキスト」という意味で用いる。

3. 日本語におけるテキスト合成演算の実現

式(1)として定式化したテキスト合成演算を機械的に実現するためには、X, R, Y, C, G の具体的内容と、列挙した条件をどのようにして満たすべきかを具体的に決定する必要がある。まず、G を除く4つについて、その内容を具体的に決定する。

テキスト X と Y　X と Y は、それぞれ文字列として表現された日本語の自然言語テキストとする。

関係 R　次の5個の関係を設定する [2,5,6]。

(1)　因果関係：Y は X のためである。X なので Y。
(2)　逆接：X、しかし Y。
(3)　時間経過：X の後で Y。Y する前に X。
(4)　対比：X、一方 Y。
(5)　共通要素を持つ：X 中に現れる表現 x_i と Y 中に現れる表現 y_j が、同一のもの e_k を指し示す。

外部制約 C　比較的簡単に判定できる次の3種類を考える。

(1)　文数の指定

(2) 文字数の指定

(3) 主題の指定

次に、出力 Z が満たすべき条件を満足するような演算を、どのように機械的に実現するかを検討する。

条件 (b)-(d) を満たすための直接的な方法は、それぞれの文をある種の意味表現に変換し、その意味表現の空間で条件 (b)-(d) を満たすような合成演算を定義する方法である。対象領域を限定すれば、原理的にはこのような方法が可能であり、意味を考慮した合成が可能となる。しかしながら、現時点では、精度の高い意味解析はそれほど期待できないため、意味解析ができない場合でも実行できる別の手段があるのが望ましい。

実現可能という面から考えるならば、日本語解析において、現在の技術で実用的な精度で実行できる処理は、構文と一部の意味情報を利用した処理 (以下「構文的操作」と呼ぶ) である。構文的操作には、構文解析や文節係り受け構造に対する変形操作などが含まれる。

文字列として表現された文は、変更しない限り、それが表現する意味内容を保存する。また、いくつかの構文的変形操作は、文が持つ意味内容をほぼそのまま保存する。入力 X、Y に対する操作を、このような構文的操作の範囲にとどめれば、X と Y が表現する意味内容は、ほぼそのまま保存することができる。

Z からは関係 R も読みとることができる必要がある。任意のテキストから関係 R を読みとれるかどうかを機械的に判断することは困難だが、1 つの関係 R に対して、その関係を読みとることができるような構文構造や文章構造を定義することは比較的容易である。

以上から、意味解析が利用できない場合の実現方法を、次のようにまとめることができる。

- それぞれの関係 R に対して、その関係が読みとれるような構文・文章構造を定義する。
- 「X に Y を関係 R で追加する」際に、関係 R に対して定義された構文・文章構造に X と Y を代入して、テキスト合成を行う。
- そのままの形で代入できない場合は、X および Y に対して、意味を保存する構文的操作を最小限適用して変形したのち、代入する。

最後に残ったのが、指針 G である。上記の方法をとる場合、文章を構成するそれぞれの文が構文的・意味的に正しいことは、近似的に成り立つと考えて良い。また、文章の結束性も、関係 R に対して定義される構文・文章構造によって近似的に保証できる。残された条件は、代名詞の使用、省略などの文章の自然さについての条件である。これらについては、上記の方法による合成結果に対して、それらをチェックし修正する規則を用意することによって実現する。

4. テキスト合成システム

前節で検討した日本語のテキスト合成を実現するシステムについて述べる。

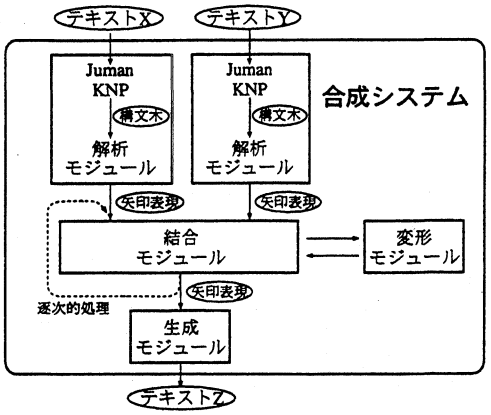

図 1 システムの構成図

4.1 システムの構成

作成したシステムは、次の 4 つのモジュールから構成される (図 1)。

(1) **解析モジュール**(ana) : 入力テキストを解析し、**矢印表現**と呼ばれる内部表現に変換する。

(2) **結合モジュール**(con) : 2 つの矢印表現を結合し、1 つの矢印表現を生成する。

(3) **変形モジュール**(trans) : 矢印表現の変形を行う。結合モジュールから必要に応じて呼び出される。

(4) **生成モジュール**(gen) : 矢印表現から、表層文を生成する。

なお、本システムでは、自然言語テキストであるための指針 G は入力として明示的に扱わず、焦点の流れと代名詞の使用についての指針をシステムの内部に組み込んだ。上記のモジュール構成は、(4 引数の) テキスト合成演算を次のように計算することを意味する。

$$Z = add\text{-}on(X, R, Y, C)$$
$$= \mathrm{gen}(\mathrm{con}(\mathrm{ana}(X), R, \mathrm{ana}(Y)), C)$$

4.2 矢印表現

本システムでは、意味表現に基づく合成と構文的操作に基づく合成の両方を実現する。これを可能とするために、表層表現レベル (文節依存構造) と意味表現レベルの両方での合成が可能な表現形式—矢印表現—を設計し、システムの内部表現として採用した。

矢印表現は、文節依存構造と文節間の意味関係を木構造で表現したものである。節点は文節依存構造を表しており、枝にはラベルと方向がついている。たとえば、「Juman は 1992 年に開発された」という文は、解析のレベルに応じて、

$$[\text{Juman}] \xrightarrow{\text{は}} [\text{ 開発された }] \xleftarrow{\text{に}} [1992\text{ 年 }]$$

$$[\text{Juman}] \xleftarrow[\text{prop(歴史)}]{} [1992\text{ 年に開発された }]$$

などの矢印表現として表現される。同様に、「Juman は

表 1　構文ラベルと意味ラベル

構文ラベル		意味ラベル	
表層格	は, が, その他格助詞	包含関係	class, instance
修飾関係	連体修飾	property	歴史, 場所, 機能

形態素解析システムのひとつだ」は、

$$[\text{Juman}] \xrightarrow{\text{は}} [\text{形態素解析システムのひとつだ}]$$

$$[\text{Juman}] \xleftarrow[\text{class}]{} [\text{形態素解析システム}]$$

として表現される。矢印表現の枝についているラベルと方向を合わせて**矢印演算子**と呼ぶ。たとえば上の例の場合、$\xrightarrow{\text{は}}$ や $\xleftarrow[\text{class}]{}$ が矢印演算子である。矢印の上につくラベルを**構文ラベル**と呼び、矢印の下につくラベルを**意味ラベル**と呼ぶ。構文ラベルは表層格や修飾関係を表しており、意味ラベルは包含関係や property(属性) を表している。現在、システムで用いている構文／意味ラベルを表 1 に示す。

4.3 解析モジュール

解析モジュールは、文字列として与えられるテキストを解析し、それに対応する矢印表現を生成するモジュールである。本モジュールでは、まず、文字列として与えられたテキストを、Juman と KNP を用いて形態素解析・構文解析し、文節依存構造に変換する。次に、得られた文節依存構造を矢印表現に変換する。文節依存構造に対する標準的な矢印表現は、以下の方法で生成する。

(1) 文節の最後の形態素が格助詞であれば、その文節とそれ以降の部分の 2 つのノードに分け、格助詞を構文ラベルとして矢印演算子でつなぐ

(2) 「動詞基本形＋名詞」、「動詞タ形＋名詞」の並びがあれば、動詞と名詞でノードを分け、構文ラベル「連体修飾」の矢印演算子でつなぐ

また、構文解析の結果として得られた文節依存構造が、意味解析用の規則とマッチする場合は、上記の標準的な矢印表現ではなく、その規則によって生成される矢印表現を出力する。現在のシステムでは、ある種の説明文に対する意味解析規則を解析テーブルの形で実装している。

4.4 結合モジュール

結合モジュールは、テキスト X とテキスト Y に対応する 2 つの矢印表現 (X_a と Y_a) と関係 R を入力とし、それらを結合して 1 つの矢印表現を出力する。実際にどのような矢印表現を出力するかは、結合規則によって決定する。3 節に示した関係 R のうちの (1) から (4) の場合は、X_a と Y_a をそれぞれダミーノードと呼ばれる特別な節点に埋め込み、その 2 つのダミーノード間に関係 R を設定した矢印表現を作成する (関係 R に対する構文・文章構造の決定は、生成モジュールによって行なわれる)。一方、2 つのテキストが共通要素を持つ場合 (5) は、4 つの結合規則のいずれかを用いて結合する。このうち、2 つは、構文レベルの結合規則 (主題 (ハ格) が共通の場合に残りの部分をマージする、一方を連体修飾句として埋め込む) であり、残りの 2 つは、意味レベルの結合規則である。

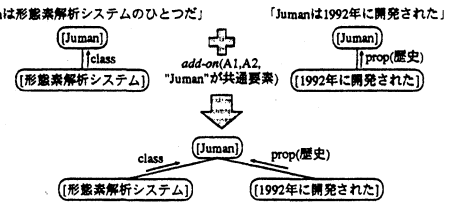

図 2　結合規則の実行例

4.5 変形モジュール

結合モジュールが 2 つの矢印表現をそのままの形で結合できないと判断したとき、もしくは制約 C で指定された主題に変更する必要があるとき、変形モジュールが呼び出される。変形モジュールは、変形規則を矢印表現に対して適用し、その結果を結合モジュールに返す。現在のシステムでは、指定した単語を根とするように矢印表現を変形する規則と、包含関係 (class-instance) の親子関係を入れ換える規則、の 2 つを実装している。

4.6 生成モジュール

生成モジュールは、矢印表現を文字列に変換する。まず、矢印表現 (木構造) を走査して、どの節点から表層化 (文字列化) するか決定する。表層化は、葉から根の方向に行なうため、ある節点が 2 つ以上の子木を持つ場合のみ、どの子木を優先して表層化するかを決定すれば、表層化する節点の順序が全て決定する。次に、その順序に従って、その節点と親節点およびその間の矢印演算子を入力として、その部分を文字列化する。その後、親節点の内容をこの文字列で置き換える。この操作により、対象節点が親節点にマージされ、消滅することになる。この操作を繰り返し適用することにより、最終的に、根節点だけの矢印表現となり、その節点内に、表層文字列が得られる。なお、それぞれの文字列化の過程で、活用形の変換や代名詞化の処理も行なっている。

本システムで実現するテキスト合成演算は、「X に関係 R で Y を追加する」ものであり、X をより重要な中心情報として扱う。日本語の文では、重要な情報は文の最後で述べられることが多い。また、文章においては、最初の文または最後の文で最も重要なことが述べられる傾向がある。そこで、

- 合成結果が 1 文の場合:
  X の末尾の文節が、合成結果の末尾の出現する
- 合成結果が 2 文以上の場合:
  X の末尾の文節が、合成結果の 1 文目の末尾に出現する

ように、表層化の順序を決定している。

5. 合 成 例

次の 2 文を「Juman が共通要素」という関係で「1 文」

表 2 合成例

X	Y	R	Z
「太郎は昨晩夜更しした」	「太郎は今朝寝坊した」	関係「結果」	「太郎は昨晩夜更ししたので、今朝寝坊した」
「形態素解析システムは文を単語に分割する」	「Juman は形態素解析システムだ」	共通要素「形態素解析システム」	「形態素解析システム Juman は文を単語に分割する」
「Juman は文を単語に分割する」	「Juman は形態素解析システムだ」	共通要素「Juman」	「Juman は形態素解析システムで文を単語に分割する」
「Juman は 1992 年に開発された」 X'	「Juman は形態素解析システムのひとつだ」 「形態素解析システムは文を単語に分割する」	共通要素「Juman」 共通要素「形態素解析システム」	「形態素解析システム Juman は 1992 年に開発された」($=X'$) 「Juman は文を単語に分割する形態素解析システムのひとつで 1992 年に開発された」
「Juman は形態素解析システムのひとつだ」 X''	「Juman は 1992 年に開発された」 「形態素解析システムは文を単語に分割する」	共通要素「Juman」 共通要素「形態素解析システム」	「1992 年に開発された Juman は形態素解析システムのひとつだ」($=X''$) 「1992 年に開発された Juman は文を単語に分割する形態素解析システムのひとつだ」

として合成する場合の合成過程を示す。

X =「Juman は 1992 年に開発された」

Y =「Juman は形態素解析システムのひとつだ」

まず、解析モジュールは、これらの文を次のような矢印表現に変換する。

$$X_a = [\text{Juman}] \xleftarrow[\text{prop(歴史)}]{} [\text{1992 年に開発された}]$$

$$Y_a = [\text{Juman}] \xleftarrow[\text{class}]{} [\text{形態素解析システム}]$$

次に、結合モジュールは、これらの矢印表現を結合し、1 つ矢印表現にまとめる (図 2)。

最後に、生成モジュールがこの矢印表現を表層化する。X に Y を追加する場合には、X を中心情報とするため、先に "class" の枝の方を表層化する。ここでは、ラベルが "class" であり、制約が「1 文」であるため、「Juman」と「形態素解析システム」を名詞句としてまとめ、

$$[\text{形態素解析システム Juman}] \xleftarrow[\text{prop(歴史)}]{} [\text{1992 年に開発された}]$$

とする。さらに "prop(歴史)" の枝を表層化し、最終的に次の文を生成する。

Z =「形態素解析システム Juman は 1992 年に開発された」

いくつかの合成例を表 2 に示す。このように、入力が短い文であれば、システムは妥当な合成結果を出力できる。

6. 重要度を考慮した逐次的合成

本システムの基本的な処理は、2 つのテキストから 1 つのテキストを合成する処理であるが、これを繰り返し逐次的に適用することにより、段階的に文を追加し、より複雑な文や文章を生成させることができる。逐次的生成を行う場合は、入力される順番が早いほど重要度が高いものとして合成を行う。つまり、同じテキスト集合を入力しても、入力の順序が異なれば、合成結果も異なる。表 2 の 3 番目と 4 番目の例は、入力順序の違いによる合成結果の違いを示している。

7. ま と め

本論文では、関連する 2 つの文からそれを組み合わせたテキストを作り出す操作をテキスト合成演算として定義し、日本語に対するテキスト合成演算を機械的に実現するプロトタイプシステムについて報告した。言い換え [3] があるテキストを別のテキストに変換する、1 入力 1 出力のテキスト演算であるのに対し、ここで提案したテキスト合成演算は、2 つのテキストを 1 つのテキストにまとめる 2 入力 1 出力のテキスト演算である。

今回実装したシステムは、テキスト合成が機械的に実現可能であることを実証するためのものであり、各モジュールの実装は不十分であるところが多い。そのため、短い文については妥当な合成結果を出力することができるが、複雑で長い文に対してはうまく動作しないも多い。より強力な合成システムを実現するためには、今後、各モジュールの動作を決定する規則群をより詳細に整備する必要がある。

参 考 文 献

1) 日笠亘, 藤井綱貴, 黒橋禎夫: 入力質問と知識表現の柔軟なマッチングによる対話的ヘルプシステムの構築, 情報処理学会研究報告, 1999-NL-134-14 (1999).
2) 木下是雄: 理科系の作文技術, 中公新書 (1981).
3) 黒橋禎夫, 酒井康行, 鍛治伸裕: 国語辞典に基づく文章理解とパラフレーズ, 言語処理学会第 7 回年次大会 ワークショップ (2001).
4) 桜井裕, 佐藤理史: ワールドワイドウェブを利用した用語検索の実現, 情報処理学会研究報告, 2000-NL-137-4 (2000).
5) 杉原厚吉: 理科系の作文技術, 中公新書 (1994).
6) 杉原厚吉: どう書くか —理科系のための論文作法—, 共立出版 (2001).