2L5-5in2

制約付き系列ラベリングの高速解法

A Fast Method for Solving Constrained Sequence Labeling Problems

竹内 文登 *1 Fumito TAKEUCHI
西野 正彬 *2 Masaaki NISHINO
安田 宜仁 *1 Norihito YASUDA
湊 真一 *1 Shin-ichi MINATO

*1北海道大学大学院 情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University

*2日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

Sequential labeling is a task for assigning proper labels to the elements in the given input sequence and actively used in such domains as natunal language processing and DNA sequence analysis. Conventionally, probabilistic models including Hidden Markov Model have been used for this task. However,these typical formulation of probabilistic models can only deal with local relationships between elements; they cannot handle non-local relationships. Chang et.al. proposed to represent non-local relationships as constraints and showed that it yields accurate labeling. However, since conventional methods resort to approximate algorithms or integer linear programming (ILP) solvers, they does not offer any computational complexity. This paper shows that the task can be solved as longest path problem on a directed acyclic graph (DAG) under constraints posed between edges. We also show that the constrained DAG longest path problem can be solved by our recently proposed method. The beauty of the method is that it offers an upper bound of computaional time based on the size of the binary decision diagram (BDD) used for representing the constraints.

1. はじめに

系列ラベリングは与えられた系列の各要素にラベルを付与する技術であり，自然言語処理や音声認識，遺伝子系列解析などの分野で利用される [Koski 01][Voutilainen 03]．たとえば，与えられた文中の各単語に名詞や形容詞といた品詞タグを付与する品詞タグ付けは自然言語処理の基礎的な課題であり，系列ラベリングによって実現される．従来，系列ラベリングには，隠れマルコフモデル（Hidden Markov Model:HMM）や条件付き確率場（Conditional Random Field:CRF）などの確率モデルが主に用いられてきた．これらの手法は訓練データから学習した確率モデルを用いて，入力の系列への尤度が最大となるラベルを付与する．HMM や CRF は計算の効率性，モデルの頑健性の観点から局所的な関係性，すなわち系列の隣接する要素に付与するラベル間の依存関係のみを考慮する．そのため，これらのモデルでは離れた要素のラベル間の非局所的な依存関係を考慮することが困難である．Chang らは，HMM を用いた系列ラベリングにおいて，非局所的な依存関係を制約として表現し，制約を満たすラベル列の中で尤度を最大とするものを求めることで，より高精度なラベリングが可能であることを示した [Chang 12]．しかし，制約を満たす，尤度最大のラベル系列を求める問題に対する効率的な多項式時間アルゴリズムが知られておらず，そのため，Chang らは整数線形計画問題 (Integer Linear Programming:ILP) ソルバーを用いた厳密解法またはビームサーチによる近似解法を用いている．しかし，ILP ソルバーを用いる手法では計算時間を見積もれないという問題が，ビームサーチを用いる手法では精度が保証できないという問題がある．一方，近年我々は経路に含まれる辺の組合せに制約のついた有向非巡回グラフ（Directed Acyclic Graph:DAG）の最適経路を求める手法として，BDD-Consrained Search（BCS）[Nishino 15] や BDD-Constrained A* Search（BCA*）[Takeuchi 15] を提案した．HMM や CRF を用いた制約を満たす最尤ラベル系列を求める問題は，この制約つき DAG 最適経路問題として定式化可能なため，BCS や BCA*を適用することで厳密解を高速に求めることができ，かつ計算量の解析も可能である．本稿では，制約つきの系列ラベリング問題に対して，BCS，BCA*が適用可能であることを示し，実験により厳密解が現実的な計算時間で得られることを示す．

連絡先: fumito@ist.hokudai.ac.jp

2. 制約付き系列ラベリング

系列ラベリングとは，入力系列 $x = x_1 x_2 \cdots x_T \in \mathcal{X}$ に対して適切なラベル系列 $y = y_1 y_2 \cdots y_T \in \mathcal{Y}$ を求める問題である．ここで，各 y_i はラベル候補集合 $L = \{\ell_1, \ell_2, \ldots \ell_{|L|}\}$ のいずれかである．T は系列長である．$\mathcal{X}$ は入力系列集合，$\mathcal{Y}$ は出力ラベル系列集合という．系列ラベリングは評価関数を $f(x, y) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ とすると，$\hat{y} = \arg\max_y f(x, y)$ をを求める組み合せ最適化問題に帰着できる．本稿では系列ラベリングを拡張して，制約付き系列ラベリングを扱う．制約付き系列ラベリングでは，入力系列 $x \in \mathcal{X}$ に対して適切なラベル系列 $\hat{y} = \arg\max_y f(x, y)$ s.t. $\forall i\ C_i(x, y) = 0$ を決定する．ここで，$C_i : \mathcal{X} \times \mathcal{Y} \mapsto \{0, 1\}$ は i 番目の制約を表し，$\{C_1, C_2, \ldots, C_k\}$ を制約集合という．$C_i(x, y) = 1$ のとき，系列 x に対してラベル列 y を付与すると制約 C_i を違反することを示す．つまり，制約付き系列ラベリングは制約集合に含まれるすべての制約を満たすような，適切なラベリングを行うことである．

Chang らは制約つき系列ラベリングによって書誌情報抽出問題を解いている．文献の題目や著者等の情報が含まれる文から，正しく文献情報を抽出する書誌情報抽出問題は，系列ラベリングによって解くことができる．この問題に，著者ラベルなどの各ラベルが離れた位置に複数箇所に現れることはないという規則や，句読点が現れるまで書誌ラベルは変わらないといった書誌情報に存在する規則を制約として追加した制約つき系列

ラベリングを行うことで，Chang らはより高精度に情報を抽出することに成功している．

2.1 隠れマルコフモデルを用いた推論

系列ラベリングには，隠れマルコフモデル（Hidden Markov Model:HMM）や条件付き確率場（Conditional Random Field:CRF）などの確率モデルが主に用いられてきた．本節では HMM を用いた系列ラベリング手法を説明する．

HMM は複数の隠れた状態をもつ確率モデルのひとつである．以下では x_i を入力系列の i 番目の要素を表す確率変数，y_i を x_i に付与されるラベルを表現する確率変数とする．HMM は x と y の同時確率を

$$P(x,y) = P(y_1)\prod_{i=2}^{T} P(y_i|y_{i-1})\prod_{i=1}^{T} P(x_i|y_i)$$

として定める．HMM を用いた系列ラベリングでは，入力系列 x に対して y との同時確率の対数 $\log P(x,y)$ を評価関数とし，この値が最大となるような $\hat{y} = \arg\max_y P(x,y)$ を求めることでラベル系列の推定を行う．HMM を用いたスコア最大となるラベル系列の推論はビタビアルゴリズムを用いることで効率よく行うことができる．このアルゴリズムは動的計画法の一種であり，次の更新式をすべての j と ℓ に対して繰り返すことで部分系列に対する対数尤度の最大値が得られる．

$$\begin{aligned} t(1,\ell) &= \log P(y_1 = \ell) + \log P(x_1|y_1 = \ell) \\ t(j,\ell) &= \max_{\ell' \in L}\{\log P(y_j = \ell|y_{j-1} = \ell') + \\ &\qquad \log P(x_j|y_j = \ell) + t(j-1,\ell')\} \end{aligned} \tag{1}$$

ここで $t(j,\ell)$ は，$y_j = \ell$ となるような j 番目の要素までのラベル部分列に対する対数尤度の最大値を表す．更新式を適用する際に，$t(j,\ell)$ の値を求める際に用いられた $t(j-1,\ell')$ を記憶することで，すべての $t(j,\ell)$ の値を求めた後にバックトラック法を実行することで最適値を与えるラベル列 $\hat{y}$ を得ることができる．

HMM を用いて制約付き系列ラベリングを行うには，入力系列 x に対して制約を満たすようなラベル y のうち $\log P(x,y)$ が最大となる y を求める．つまり，

$$y^{\max} = \arg\max_y \log P(x,y) \text{ s.t } \forall i\ C_i(x,y) = 0 \tag{2}$$

を求める．しかし，ビタビアルゴリズムで求めたラベル系列 y は，必ずしも全ての制約 $C_i(x,y)$ に対して $C_i(x,y) = 0$ を満たすとは限らないため，ビタビアルゴリズムで制約付き系列ラベリングを行うことができない．

2.2 制約つき DAG 最長路問題

本節では，前節で述べた制約を満たすラベル系列のうち対数尤度を最大とするものを求める問題が，辺の組合せに制約がついた有向非巡回グラフ（Directed Acyclic Graph:DAG）上の最長路問題になることを説明する．

まず，ビタビアルゴリズムが探索する空間は DAG になっていることを説明する．更新式 (1) に対応して，図 1 のように $t(j,\ell_i)$ を頂点とするグラフ $G = (V,E)$ を考える．つまり，$V = \{t(j,\ell_i)|1 \le j \le T, 1 \le i \le |L|\} \cup \{s,t\}$ であり，s,t はそれぞれスタート頂点，ゴール頂点と呼ぶ．そして，更新式 (1) においての遷移の関係がある頂点間に有向辺を張る．つまり，$E = \{(u,v) \mid u = t(j-1,\ell_i), v = t(j,\ell_k), 1 \le i,k \le |L|, 2 \le j \le T\} \cup \{(s,v) \mid v = t(1,\ell_k), 1 \le k \le |L|,)\} \cup \{(u,t) \mid u = t(T,\ell_i), 1 \le i \le |L|\}$ とする．辺の重みは $\log P(y_j = \ell|y_{j-1} = \ell') + \log P(x_j|y_j = \ell)$ とする．ただし，s を始点とする辺の重みは $\log P(y_1 = \ell_i) + \log P(x_1|y_1 = \ell_i)$，$t$ を終点とする辺の重みは 0 とする．こうして作成された有向グラフには閉路が存在しないため DAG であることが分かる．ビタビアルゴリズムは，この DAG において s から t への最長路問題を解いていることに他ならない．

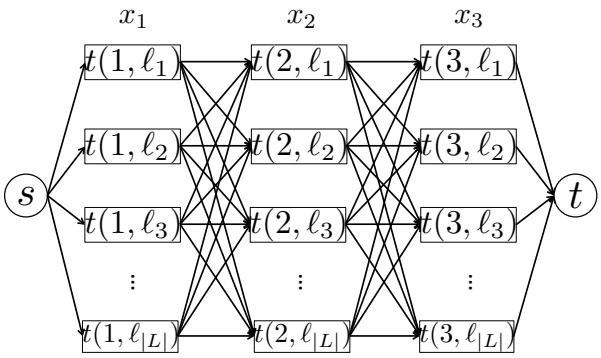

図 1: ビタビアルゴリズムにおける値の更新の様子

経路に含まれる辺の組合せ方に関する制約が存在する DAG 最長路問題を，制約つき DAG 最長路問題とよぶ．制約つき系列ラベリングにおける各制約が，図 1 の DAG 上の辺の組合せに関する制約として表現できるのであれば，制約つき DAG 最長路問題を解くことで制約つき系列ラベリングを実行できる．そして，[Chang 12] で用いられている制約など，辺の組合せに関する制約で系列ラベリングで用いられる制約条件の大半を表現可能である．そこで，以下では制約つき系列ラベリングが，辺の組合せ方に関する制約つきの DAG 最長路問題に帰着できるものとして，その効率的な解法を与える．

3. 制約つき DAG 最長路問題の高速解法

前節で述べたように，観測された系列に対する制約を満たす HMM 上の尤もらしい状態遷移を推定する問題は，制約つき DAG 最長路問題として定式化できる．近年，我々は制約つき DAG 最長路問題を解く手法として BDD-Constrained Search（BCS）と BDD-Constrained A* Search（BCA*）を提案した．本節では，BCS と BCA*のアルゴリズムの説明を行う．

BCS と BCA*はともに，図 2 のような辺重み付き DAG と Binary Decision Diagram（BDD）を入力とする．BDD は論理関数を表現するデータ構造で，DAG 上の辺に関する論理制約を表現したものである．出力は BDD が表現する制約を満たす DAG 上の最長路である．

3.1 Binary Decision Diagram

Binary Decision Diagram（BDD）は論理関数 $F : \{0,1\}^n \to \{0,1\}$ を図 2(b) のような DAG の形で表現するデータ構造である [Bryant 86]．以下では BDD の頂点を節点といい，辺を枝という．BDD は分岐節点と終端節点の 2 種類の節点をもつ．このうち，終端節点とは子をもたない節点であり，BDD は $\bot$-終端節点と $\top$-終端節点の 2 種類の終端節点をもつ．終端節点でない節点は分岐節点とよばれ，分岐節点からは必ず二本の枝が出ている．これらの枝は論理変数の値が 0 または 1 であるときに対応しており，それぞれ 0 枝（破線）と 1 枝（実線）という．BDD では，入力に応じてルート節点から

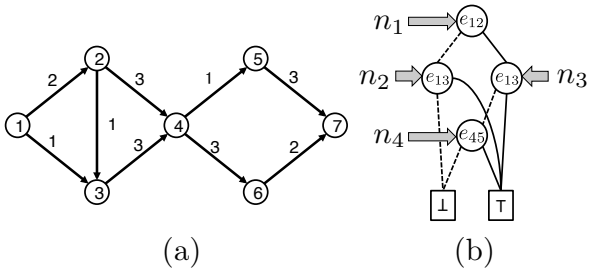

図 2: DAG と BDD の例

0-枝, 1-枝をたどって到達した終端節点によって，その入力に対する F の出力を得ることができる．すなわち，⊥-終端節点に到達したなら F は 0 を出力し，⊤-終端節点に到達したなら 1 を出力する．また BDD の各節点は部分的な論理関数に対応している．部分的な論理関数とは，ルート節点からその節点までの変数の値の割り当てを行った際の残りの変数に関する論理関数である．

BDD においてルート節点から終端節点までの経路に現れる変数の順序がすべて同じである BDD を順序付き BDD という．また，冗長な節点が現れない BDD を既約な BDD という．冗長な節点とは，0 枝と 1 枝の指す節点が等しい節点と，同じ変数に対応し同じ論理関数を表現する節点のことである．以下の節では，BDD はすべて既約な順序付き BDD であるものとする．

3.2 BDD-Constrained Search

BDD-Constrained Search（BCS）は，制約つき DAG 最長路問題を効率よく解く手法である．具体的には，BDD を用いることで，探索における「等価な状態の共有」と「枝刈り」を効率よく行うことによって，探索の状態数を削減し，処理を高速化する．「等価な状態の共有」とは，DAG 上の $s-v$ 間の経路のうち，$v-t$ 間の辺に関する制約が等しい経路のうち最長の経路のみを記憶することであり，等価な状態の共有を行うことで重複した探索を避けることができる．「枝刈り」とは，$s-v$ 間の辺の組合せが既に制約を満たさないと分かった時点で，その後の探索を打ち切ることである．BCS の詳細は [Nishino 15] を参照のこと．

BCS は探索における状態を DAG 上のノード v と BDD の節点 n のペア (v,n) によって表現する．このペアは，DAG 中のある $s-v$ 間の経路と，残りの $v-t$ 間の経路が満たすべき制約条件を表現している．この表現のもと，制約付きの最長路問題は初期状態 (s,r)（r はルート節点）から終了状態 $(t,\top)$ までのスコア最大の状態遷移の列を探す問題として表現できる．探索においては，BDD を用いることで状態遷移の計算，制約を満たさない状態の削除，および等価な状態の統合が効率的に行えるため，探索を効率化できる．具体的な手続きとしては，状態 (v,n) に到達するための最長経路を二次元配列 $T[v][n]$ に格納し，これを

$$T[v_t][n'] = \max(T[v_s][n] + w(v_s,v_t), T[v_t][n']) \qquad (3)$$

に従って動的計画法によって再帰的に更新することで解を求める．ここで，n' は BDD の節点であり，n から辺 e を通った際の v_t 以降に現れる辺に関する制約を表現する．$w(v_s,v_t)$ は辺 $e=(v_s,v_t)$ の重みである．すべての辺に対して更新式 (3) を適用したあと，$T[t][\top]$ に記憶される値が制約を満たす最長の経路長である．更新式 (3) において，$T[v_t][n']$ を決定する $T[v_s][n]$ を記憶しておくことで，バックトラック法により最長路を復元できる．

例として図 2 において，BDD で表現される制約 $F=(e_{12}\wedge e_{45})\vee e_{13}$ を満たす頂点 1 から頂点 7 までの最長路を求める．スタートの状態 $(1,n_1)$ においては $T[1][n_1]=0$ である．$(1,n_1)$ から辺 e_{12} を通ると状態 $(2,n_4)$ へ遷移する．このとき，$T[2][n_4]=2$ となる．同様に辺 e_{13} を通るとき $T[3][\top]=1$ となる．次に辺 e_{23} を通る場合を考える．頂点 2 に至る状態は $(2,n_4)$ のみであり，この状態から辺 e_{23} を通る場合は状態 $(3,n_4)$ へ遷移し，$T[3][n_4]=T[2][n_4]+w(2,3)=3$ となる．以降も同様に探索を続け，$T[7][\top]$ に記憶される値が制約を満たす最長の経路長となる．

3.3 BDD-Constrained A* Search

BCS に A* 探索を適用したものを BDD-Constrained A* Search（BCA*）という．A* 探索は最短路を求めるための，最良優先探索アルゴリズムである．BCA* は A* 探索により BCS が探索する空間をすべて探索することなく解を求めることができるため高速な探索が可能である．BCA* 探索に用いるヒューリスティック関数は，DAG 上の最長路問題を解くことで得られる DAG ヒューリスティック関数，BDD 上の最長路問題を解くことで得られる BDD ヒューリスティック関数，これら二つを同時に用いる DAG&BDD ヒューリスティック関数の三つが提案されている．本稿では紙面の都合上，DAG ヒューリスティック関数のみ説明する．BCA* の詳細は [Takeuchi 15] を参照のこと．

BCA* では，優先度付きキューを用いて次に探索する状態を決定する．BCA* のアルゴリズムはスタートの状態 (s,n_1) をキューに追加するところから始まる．次に，キューに入っている状態 (v,n) のうち，それまでの経路長 $g(v,n)$ とゴールまでの推定経路長 $h(v,n)$ を求め，その和 $f(v,n)=g(v,n)+h(v,n)$ が最も大きい状態を次に探索する．このとき，ゴールの状態 $(t,\top)$ のヒューリスティック関数の値は 0 とする．この操作をゴールの状態が展開されるまで繰り返し，そのときの $g(t,\top)$ が最長路の経路長である．

3.3.1 DAG ヒューリスティック関数

状態 (v,n) からゴール状態 $(t,\top)$ までの最長路長の推定値 $h(v,n)$ として，制約を無視した DAG 上の頂点 v から t までの最長路長 $h(v)$ を用いることができる．つまり，$h(v,n)=h(v)$ として，BCA* を行うことができる．この関数を DAG ヒューリスティック関数という．

DAG ヒューリスティック関数を用いた BCA* は，DAG 上で経路長が長い経路から順に探索を行う．もし，制約のない DAG 最長路問題の解が制約をみたすならば，BCA* はゴールの状態まで最長路に含まれる頂点のみを展開し，瞬時に最長路を得る．つまり DAG ヒューリスティック関数を用いた BCA* は，制約の有無によって最長路長が大きく変わらない問題に対して有効である．

4. 実験

4.1 実験設定

制約付き系列ラベリングに対して BCS と BCA*が現実的な時間で計算可能であることを示すための実験を行う．本実験では書誌情報抽出問題を扱う．書誌情報抽出問題とは，学術論文の参考文献欄などから著者名や論文題目などの書誌情報を抽出する問題であり，Chang らはこれを制約つき系列ラベリング

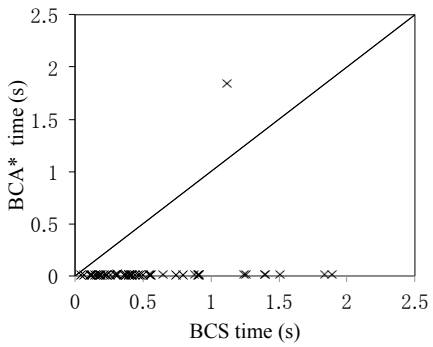

図 3: BCS と BCA* の計算時間の比較

問題として定式化して解いた．本実験では，この制約つき系列ラベリング問題を HMM を用いた制約を満たす最尤推定を行うことで解き，2.2 節で述べたように制約付き DAG 最長路問題として解く．制約は Chang らが検証で用いていた 12 種類の書誌情報抽出のための規則を，論理関数で表したものを用いる．また，HMM のパラメータは Chang らが提案した手法を用いて学習した．

実験の流れとして，まず HMM を 300 の訓練データを用いて学習した．次に学習した HMM を用いて，50 個のテストデータに対する制約を表現する BDD をそれぞれ構築し，制約満たす最尤推定を BCS，BCA*を用いて行う．この時の BDD の構築時間，BCS，BCA*の実行速度を測定する．比較実験として，幅優先探索による厳密解法とビームサーチによる近似解法により制約満たす最尤推定を行った際の計算時間を測定した．

BDD の構築と BCS，BCA* は C++で実装し，HMM の学習とビームサーチによる近似解法，幅優先探索による解法は Chang らが公開している Java による実装を用いた．実験は全て Linux64bit マシン，intel Core i7-2700K CPU，主記憶 32GB で行った．

4.2 実験結果

BCS の計算時間に対する BCA* の計算時間の散布図を図 3 に示す．幅優先探索による厳密解法では，メモリ不足により計算不可能であった．BDD の構築時間，BCS，各ヒューリスティック関数を用いた BCA* の計算時間，近似解法による計算時間の中央値を表 1 に示す．

4.3 考察

幅優先による厳密解法では探索空間が大きすぎるため，メモリ不足になったのに対し，BCS，BCA*では現実的な時間で厳密解を得ることに成功している．これは，BDD を用いた等価な状態の共有や枝刈りが有効であるからだと考えられる．BCS と BCA* の計算時間を比較すると，多くのケースで BCA* の方が高速である．しかし，BCS の方が高速になるケースも存在し，このケースでは他のケースに比べて BCA* が探索する状態数と BCS の探索する状態数と大きな差がなかった．BCA* の探索する状態数が BCS と比べて少なくならない場合，A* 探索に必要なキューの操作にかかる時間が要因で BCA* が遅くなったと考えられる．

ビームサーチによる近似解法は BCA* に対して 100 倍ほど高速に計算可能である．しかし，ビームサーチで得られる解は必ずしも厳密解ではないため，系列ラベリングの精度に保証がない．精度に関する実験は今後の課題とする．一方，厳密解を得る BCS，BCA* は近似解法に対して 100 倍程度の時間が必要であるものの，現実的な時間で計算可能である．

BDD 構築	BCS	BCA*	近似解法
2149	413	29.5	26.6

表 1: 各手法の計算時間の中央値 (ms)

BCS と BCA* の計算時間の内訳を見ると，BDD の構築時間が支配的であることが分かる．BDD の構築時間は制約に依存するが，局所的な制約ほど節点数が小さなコンパクトな BDD で表現でき，大域的な制約ほど節点数の大きくなる傾向がある．今回用いた制約のち，各ラベルは離れ箇所に 2 箇所以上現れてはならないという制約があり，これは大域的な制約であり，この制約を表現する部分の構築にかかる時間が BDD 構築にかかる時間の大部分を占めた．より高速に BDD を構築する方法が望まれるが，これは今後の課題である．

5. 結論

本稿では制約付き系列ラベリングに BCS と BCA* が有効であることを示した．具体的には，HMM や CRF を用いた最尤推定を行う際に，系列の構造を制約として表現し，この制約を満たすような尤度最大のラベリングが可能であることを示した．そして実験から BCS，BCA* を用いることで現実的な計算時間で厳密解を求めることができることを示した．特に，BCA* を用いることでより高速に解を求めることに成功した．

謝辞

本研究の一部は科研費基盤 (S)15H05711 の助成による．

参考文献

[Bryant 86] Bryant, R. E.: Graph-based algorithms for boolean function manipulation, *Computers, IEEE Trans. on*, Vol. 100, No. 8, pp. 677–691 (1986)

[Chang 12] Chang, M.-W., Ratinov, L., and Roth, D.: Structured Learning with Constrained Conditional Models, *Mach. Learn.*, Vol. 88, No. 3, pp. 399–431 (2012)

[Koski 01] Koski, T.: *Hidden Markov models for bioinformatics*, Computational biology, Kluwer Academic Publishers Norwell, MA, Dordrecht, Boston (2001)

[Nishino 15] Nishino, M., Yasuda, N., Minato, S., and Nagata, M.: BDD-Constrained Search: A Unified Approach to Constrained Shortest Path Problems, in *Proc. of AAAI*, pp. 1219–1225 (2015)

[Takeuchi 15] Takeuchi, F., Nishino, M., Yasuda, N., Akiba, T., Minato, S., and Nagata, M.: A Fast Method for Solving Constrained Shortest Path Problems on Directed Acyclic Graphs, *IEICE technical report*, Vol. 115, No. 344, pp. 9–16 (2015)

[Voutilainen 03] Voutilainen, A.: Part-of-speech tagging, *The Oxford handbook of computational linguistics*, pp. 219–232 (2003)