研究で開発したコードの公開

吉永 直樹
東京大学 生産技術研究所
ynaga at tkl.iis.u-tokyo.ac.jp
http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/

謝辞

本資料を作成するにあたって，楽天技術研究所の新里圭司氏には，企業研究者の立場から貴重な意見を頂きました.

また，喜連川・豊田研究室の自然言語処理班メンバーには，周辺分野研究者，あるいは学生としての立場から，貴重な意見を頂きました.
合わせてここに感謝いたします.

病床にありながら私をいつも精神的に支え続けてくれた妻・知恵子と，リンクに感謝します.

本章の流れ

- 研究者がコードを公開する意義
- 二種類のコード: 実験コードとソフトウェア
- 実験コードを公開するには
- 実験コードをソフトウェア化する
- 日本語係り受け解析器 J.DepP の開発

研究で開発したコードを公開する意義

- 実験結果の再現性を保証 [Sonnenburg+ '07]
  - 提案手法の透明性・信頼性を確保
  - 再実装のコストを削減し，比較を容易に

  ref. http://www.cicling.org/2011/right.htm#Verifiability

- 研究成果の社会への還元
  - 研究成果をツールとして共有資産化
    - 基礎解析(品詞・構文・意味解析), 知識獲得
  - 研究成果を一般ユーザに提供（デモでも可）
    - 応用タスク（機械翻訳・評判分析など）

コードを公開しないとどうなる？

- 実験結果の再現が実質的に困難に
  [Bod '96, Bikel '04, Loosi & Canu '07]
  - 論文には手法の細部までは書けない
  - 細部の実装は実装者の裁量に任される

- 特定のデータセットに基づく局所的な数値評価に依存するように

  効率的な研究開発の背後に大きなリスクが存在
  - ドメイン依存性 [Gildea '01, Petrov '10]
  - 注釈の曖昧性・誤り [Manning '11]
  - オーバーヘッドの軽視 [Tsuruoka '04]

コードを公開するのは難しい？

- 2011年以降，言語処理分野の国際会議ではコード・データを募集するようになった
  - CICLING, ACL, EMNLP, EACL, COLING, ...

	2011	2012
ACL	17 / 292	2 / 187
EMNLP	6 / 149	4 / 139

（ACL Anthology より）

- 論文と同時に提出されるコードは少ない
  - コードを出せない，まとめる余裕がない
  - 採否に影響がなさそうだったのでやめた？

本章の流れ

- 研究者がコードを公開する意義
- 二種類のコード: 実験コードとソフトウェア
- 実験コードを公開するには
- 実験コードをソフトウェア化する
- 日本語係り受け解析器 J.DepP の開発

コードを公開する目的を意識する

- 実験コード（参照実装）[Lin '12]
  - 目的: 手法の透明性，実験結果の再現性を確保
  - （主に）研究者による利用を想定

- ソフトウェア
  - 目的: 汎用性のあるツールとしての提供
  - 非研究者を含む一般ユーザによる利用を想定

まずは実験コードとしての公開を検討し，
需要に応じてソフトウェア化する

本章の流れ

- 研究者がコードを公開する意義
- 二種類のコード: 実験コードとソフトウェア
- 実験コードを公開するには
- 実験コードをソフトウェア化するには
- 日本語係り受け解析器 J.DepP の開発

実験コードの公開の流れ

- 公開するコードをまとめる
  - ファイル単位でコードの依存関係を整理
  - コードの整理は必要最低限に
- README（実験結果の再現方法）を書く
  - インストール方法，入出力の形式，実行ログ
  - 使用条件に関する断り書き
- 研究（手法）と対応づけて公開する
  - 論文に添付して投稿（または入手方法を記載）
  - 論文リストに添書き

実験コードの公開ができない方へ (1/2)

- 異動・卒業により研究から離れる（学生）
- コードの著作権利者が自分だけではない
  - 第三者の（非公開）コードに対する追加実装
- 特殊な実行環境が要求される
  - 計算機環境（並列分散環境・膨大なメモリ）
  - 商用ライブラリ，専有データ（クエリログ等）
- 知財の絡みで公開が難しい（企業研究者）
  - コードは資産（商用利用する）

実験コードの公開ができない方へ (2/2)

- 共著者にコード・README を引き継ぐ
- 再実験を可能にする別手段を提供する
  - バイナリ形式での公開
  - デモ・実行サーバの提供
  - 受け取ったデータに対して代理実行
- 再実装の助けとなる情報を提供する
  - 標準データセットでの実験結果を差分で公開
    例: 品詞タグ付け→文ID + 品詞タグ列
  - 実装の詳細を別資料として添付

実験コードの公開をためらう方へ

- 公開したいけどソフトウェアとして未完成...
  → 実験コードと割りきって公開しよう
  - 実験コードは実験結果が再現できれば必要十分
  - 非公開＝比較のコスト発生（引用されにくい）

- （万が一）バグが見つかったらどうしよう...
  → 自信が無いコードこそ積極的に公開すべし
  - バグを含むコードを用いた実験の再現性は？
  - 第三者による検証を通して研究の信頼性を担保

そのコードで削減される時間があることを意識する

本章の流れ

- 研究者がコードを公開する意義
- 二種類のコード: 実験コードとソフトウェア
- 実験コードを公開するには
- 実験コードをソフトウェア化するには
- 日本語係り受け解析器 J.DepP の開発

ソフトウェア化を検討する

- ソフトウェア化に値する手法はごく一部
  - 競合ソフトウェアに対する優位性があるか？
  - 研究における評価基準以外の価値観が存在

- 実験コード（研究）は一点突破
  - 言語処理では出力の質（精度）に最大の関心
  - 論文で評価しない性能には目をつぶることも

- ソフトウェアは多様な価値観に晒される
  - 1%の精度向上で何が具体的に変わるのか？
  - 応用ごとに異なる価値観が存在（精度＜速度）

実験コードからソフトウェアへ

- 名前を決める
- ウェブサイト・ドキュメントの作成
- 入出力の形式（フォーマット）を整理
- （コマンドライン）オプションの整備
- ライセンスの決定
- コードの整理
- バージョン管理システムへの登録
- SNS等を利用して公開を宣伝

ソフトウェア化する上での基本方針

- ユーザに直接触れるところを重点的に整備
  - 名前・ウェブサイト
  - 入出力の形式・オプション

- 迷ったら競合ソフトウェアに合わせる
  - 標準的な入出力の形式を知る
  - オプションからユーザが使う文脈を学ぶ
  - 移行・比較・テストを容易に

変える必要のないところは変えず**価値がある差**を付けられるところに時間をかける

名前を決めてウェブサイトをつくる

- ソフトウェアの名前
  - 他ソフトウェアとの区別
  - 名は体を表す: 用途を明確化（Malt*Parser*）

  講演者の場合: 原則頭字語ベース
  - **J.DepP**: **J**apanese **Dep**endency **P**arsers
  - **opal**: **o**nline **p**assive-**a**ggressive **al**gorithm
  - **yakmo**: **y**et **a**nother **k**-**m**eans via **o**rthogonalization
  - ...

    opal は一般的過ぎて失敗（追跡できない）

- ウェブサイトの整備
  - 特徴，API，更新履歴，ベンチマーク，実行例

オプションを整理する

- -h, --help で必要十分な Usage を出す

  多くのユーザはドキュメントを読まない

  - ユーザ視点で自然なオプション書式を採用
    （GNU Getopt など; 独自書式は避ける）
  - オプションはユーザの使用頻度が高い順に

- オプションのデフォルト値を吟味する

  多くのユーザはデフォルトの設定でしか使わない

  - 調整パラメタが多いと困惑（極力減らす）
  - 特定の入力に依存しない値に設定

入出力の形式を決める

- 標準的な入出力形式をサポートする
  競合ソフトウェア・標準データセット(CoNLL等)
  - 移行・比較・テストを容易に
  - 標準的な形式がなければ応用を意識して設計
  - machine-friendly / human-friendly な出力を用意

- J.DepP（日本語係り受け解析器）の場合
  - 入力: 生文，JUMAN / MeCab の出力
  - 出力: 係り受け（CaboCha / KNP 互換）

```
> mecab < text | jdepp > parsed
> jdepp < text > parsed
> jdepp < text | to_tree.py
```

ライセンスを選択する

- ライセンスの目的
  - 開発者の権利(著作権等)を保護する
  - 利用者の権利(複製・改変等)を明らかにする
- 経験則: 既存の OSS ライセンスから選ぶ
  GPL, LGPL, BSD,.. (http://opensource.org/licenses)
  - コーナーケースに関するノウハウが蓄積
  - 選び方については [Sonnenburg+ '07] を参考に
- 独自ライセンスはなるべく避ける
  - 解釈を巡り問題となる場合も: ICOT 条項

    ref. http://sourceforge.jp/projects/naist-jdic/

コードを整理する

- コード上・仕様上のバグ・制限を取り除く
  - ハードコードされた入力依存の定数を変数に
  - メモリリークの撲滅 (valgrind)
  - 想定外の入力に備える

  ソフトウェアテストは工藤さんのトークで

- コードの完成度を上げる（C++の場合）
  - コンパイラの警告への対処

    (g++ *-Wall -Weffc++*, clang++ *-Weverything*)
  - 静的解析ツールによる検証 (cppcheck)
  - 変数・関数名の可読性の改善

  局所的な改善に陥らないよう留意する

バージョン管理システムへの登録

- ソフトウェアではバージョン管理は必須
  Subversion, Git, Mercurial, Bazaar...
  - 論文投稿時のコード（実験コード）を保存
  - デバッグへの活用（バージョン間で差分確認）
  - （第三者の）実験の再現性に関わる

- OSS ホスティングサービスとの連携
  GitHub, Google Code, SourceForge, ...
  - 公開用ウェブサイトのひな形を Wiki で提供
  - 安定版と開発版（バグ修正・試験実装）の管理
  - ダウンロード数の把握

本章の流れ

- 研究者がコードを公開する意義
- 二種類のコード: 実験コードとソフトウェア
- 実験コードを公開するには
- 実験コードをソフトウェア化するには
- 日本語係り受け解析器 J.DepP の開発

J.DepP とは

- 統計的日本語係り受け解析器 (2009年12月公開)
  http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/
  入力:(形態素解析済の)文
  出力: 文節区切り＋文節間係り受け

  - 主要な解析アルゴリズム([Sassano '04]等)を実装
  - 主要な形態素解析器(辞書)をサポート:
    JUMAN, MeCab (JUMAN, IPA, UniDic 辞書)
  - 精度 ~92% (学習~30秒); 速度 >10000文／秒
    [ver. 2013-01-23; MacBook Air (Mid 2011)]

J.DepP 開発の背景

- 言語処理では品詞解析と構文解析の間に壁
  - 品詞解析: (精度) ~97%, (速度) >10000文／秒
  - 構文解析: (精度) ~90%, (速度) ~100文／秒

    ref. [Pantel '04, 辻井 '10]

- 研究で扱う「テキスト」と実世界テキストの乖離
  - 研究では新聞記事での解析精度の評価に関心
  - 新聞（数千文／日）vs. twitter (数千万文／日)

    ref. 毎日新聞: http://www.nichigai.co.jp/sales/corpus.html

    tweet counter: http://hide.dyndns.info/tweetcounter/

目標: 最高精度の構文解析を品詞解析まで速く

J.DepP の開発から生まれた研究 (1/2)

- 2008年夏: プロトタイプ実装
  - 解析アルゴリズム [Sassano '04] は速い
  - 大量の組み合わせ素性を考慮した分類が遅い
- 組み合わせ素性を用いた分類の高速化
  [Yoshinaga & Kitsuregawa '09]
  - 査読コメント: ... tackles **a problem that is often overlooked but has a large impact on the actual runtime of many NLP processors**.
- 分類器を実験コードとして公開 (pecco)
  - 上位下位関係抽出ツールへの転用
    http://alaginrc.nict.go.jp/hyponymy/

J.DepP の開発から生まれた研究 (2/2)

- 2009年末公開: 解析モデルは同梱せず
  - 理由: コーパスの著作権上，同梱にリスク
  - モデルの学習には膨大な時間が必要
- 構成的カーネル計算を用いた学習の高速化
  [Yoshinaga & Kitsuregawa '10]
  - 訓練時間: >8時間 (SVM)→33秒 (メモリ: <100MB)
    コンパイル時のユーザサイドでの学習を現実的に
  - 「学習→テスト」のサイクルを効率化＝高精度化
- 学習器を実験コードとして公開 (opal)

J.DepP を公開してからの紆余曲折 (1/5)

- ユーザから質問・要望が届く
  - コンパイルできない，インストールできない
    ビルドツール (GNU autotools) への対応で対処
  - JUMAN，MeCab/naist-jdic と連携できないか？
    学習コーパス変換スクリプトを追加し対応
  - 商用利用できる解析モデルは無いか？
    KNB コーパスからの学習に対応
  - ライセンスを LGPL にして欲しい
    GPL/LGPLのデュアルライセンスに
  - …

J.DepP を公開してからの紆余曲折 (2/5)

- ユーザからフィードバックが届く
  - 日本語 NLP における因習でつまづくユーザ
    - 文を区切らず入力（汎用文分割ツールなし）
    - 半角文字を入力（コーパスでは全角に正規化）

    入力の正規化（整形）が解析結果に大きく影響

  - 融通が効かない教師あり学習ベースの解析
    - 彼に | メールする vs. 彼へ | メール | する
    - 2，580円の「，」の後で文節が切れる

    ユーザサイドで出力を調整することが困難

J.DepP を公開してからの紆余曲折 (3/5)

- 実装の一部に関して特許が出願されているとの指摘 (2010年7月)
  →開発者に確認して解決
  We do not guarantee that the implemented algorithms other than those proposed by us are patent-free と追記
- 一般には開発者から特許について指摘される場合も
  - Bayesian Setsの特許について
    http://d.hatena.ne.jp/mjmania/20100325/
  - Compressed Permuterm Index | Preferred Research
    http://research.preferred.jp/2012/11/comp_perm_index/
    （研究・非営利目的の利用に限定して OSS 化が許可される）

J.DepP を公開してからの紆余曲折 (4/5)

- 効率を重視した開発方針が裏目に出る
  - スクリプト言語による実装
    - 仕様が不安定＝移植性が低い
    - 実行速度が本質的に遅い（C/C++比）
  - 外部ライブラリへの（過度の）依存
    - Boost, *hash, Intel TBB, Eigen, etc.

  パッケージ管理システムへの登録により対処可能

- 実験コードに含まれないコードの品質管理
  - 提案手法による貢献以外の所にボトルネック
  - 入出力の整形，前処理，素性抽出，I/O など

J.DepP を公開してからの紆余曲折 (5/5)

- ソフトウェアと実験コードの乖離
  - ソフトウェアは動的（継続的に改良・更新）
  - 実験コードは静的

  J.DepP の場合 (2009-12-01→2013-01-23)
  - 解析精度の改善 (素性工学): 90.93% → 92.09%
  - 解析速度の向上 (実装の改善): データ構造 (x2), 枝刈り (x1.5), SIMD 命令 (x1.5) ...

  論文を引用してもらいたいが，厳密には
  対応しない（実験コードではない）

（それでも）ソフトウェアを公開して良かった

- 他者に自分の研究成果が使われる喜び
  - ソフトウェアによる貢献＞論文で報告する知見
  - ソフトウェアが利用される＝研究が評価される
- 第三者による開発への支援活動
  - 実データで得られた知見のフィードバック
  - パッケージ管理システムへの登録
  - バグ報告・パッチの提供
- 研究業界における知名度の向上

本章のまとめ

- 論文を書いたらコードの公開を検討すべし
  - 手法の透明性・実験の再現性を保証
  - 公開できない場合は代替手段を検討

  捨てられてしまうそのコードに陽の目を

- ソフトウェアは研究者が社会と繋がる手段
  - さまざまなユーザからフィードバック
  - 実応用を通して提案手法の相対的価値を把握

  多くの人に使われてこそ技術は意味を持つ

参考文献

S. Sonnenburg, M. L. Braun, C. S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.-R. Müller, F. Pereira, C. E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, R. Williamson. **The need for open source software in machine learning.** JMLR 8(Oct), 2443–2466. 2007.
http://www.jmlr.org/papers/volume8/sonnenburg07a/sonnenburg07a.pdf

D. Gildea. **Corpus Variation and Parser Performance.** Proceedings of EMNLP. 2001.
http://www.aclweb.org/anthology-new/W/W01/W01-0521.pdf

S. Petrov, P.-C. Chang, M. Ringgaard, and H. Alshawi. **Uptraining for Accurate Deterministic Question Parsing.** Proceedings of EMNLP. pp. 705--713. 2010.
http://aclweb.org/anthology-new/D/D10/D10-1069.pdf

C. D. Manning. 2011. **Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?** CICLing 2011, Proceedings, Part I. pp. 171--189.. http://nlp.stanford.edu/~manning/papers/CICLing2011-manning-tagging.pdf

Y. Tsuruoka, and J. Tsujii. **Iterative CKY parsing for Probabilistic Context-Free Grammars. Proceedings of IJCNLP.** pp. 52-60, 2005.
http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/papers/ijcnlp04.pdf

R. Bod. **Efficient Algorithms for Parsing the DOP Model ? A Reply to Joshua Goodman.** cmp-lg/9605031. http://arxiv.org/pdf/cmp-lg/9605031

G. Loosli and S. Canu. Comments on the "Core Vector Machines: Fast SVM Training on Very Large Data Sets". JMLR 8 (Feb): 291--301. 2007. http://www.jmlr.org/papers/volume8/loosli07a/loosli07a.pdf

D. M. Bikel. **Intricacies of Collins' Parsing Model.** Comp. Ling. 30 (1). 2004 http://acl.ldc.upenn.edu/J/J04/J04-4004.pdf

Chih-Jen Lin. **Machine learning software: design and practical use** (MLSS 2012 tutorial): http://www.csie.ntu.edu.tw/~cjlin/talks/mlss_kyoto.pdf

P. Pantel, D. Ravichandran, and E. Hovy. **Towards Terascale Knowledge Acquisition**. Proceedings COLING. pp. 771-777. 2004. http://aclweb.org/anthology-new/C/C04/C04-1111.pdf

M. Sassano. **Linear-time dependency analysis for Japanese.** Proceedings of COLING. pp. 8--14. 2004. http://aclweb.org/anthology-new/C/C04/C04-1002.pdf

辻井潤一. **言語理解と知識 ー情報空間の構造化に向けてー.** 人工知能学会全国大会2010 基調講演. http://kaigi.org/jsai/webprogram/2010/pdf/keynote_tsujii.pdf

N. Yoshinaga and M. Kitsuregawa. **Polynomial to Linear: Efficient Classification with Conjunctive Features.** Proceedings of EMNLP, pp. 1542--1551. 2009. http://aclweb.org/anthology-new/D/D09/D09-1160.pdf

N. Yoshinaga and M. Kitsuregawa. **Kernel Slicing: Scalable Online Training with Conjunctive Features.** Proceedings of COLING (oral), pp. 1245--1253. 2010. http://aclweb.org/anthology-new/C/C10/C10-1140.pdf

参考文献（適宜追加）

A. Fokkens, M. van Erp, M. Postma, T. Pedersen, P. Vossen, and N. Freire. **Offspring from Reproduction Problems: What Replication Failure Teaches Us: WordNet.** Proceedings of ACL. pp. 1691--1701. 2013.
　既存手法の再実装を通して得られた知見・教訓をまとめた論文

T. Pedersen. **How would I like to see ACL conferences develop and change in the next fige years?** Talk given at ACL 2011 business meeting. 2011.
http://www.slideshare.net/duluthted/pedersen-acl2011businessmeeting
　上記論文の著者による，実験の再現性に関する議論