УДК 004.021

СОВЕРШЕНСТВОВАНИЕ АЛГОРИТМА OSTIA ДЛЯ ОБУЧЕНИЯ ТРАНСДУКТОРОВ

Э.В. Малахов, О.М. Замятина

Томский политехнический университет
E-mail: malakhov@tpu.ru

На примере алгоритма OSTIA рассмотрена задача обучения трансдукторов (двухленточных автоматов) на конечном множестве пар, задающих цепочку входного языка и эквивалентную ей цепочку выходного языка. Предложен метод использования множества цепочек, заведомо не входящих во входной язык, с целью уменьшения погрешности обучения по алгоритму OSTIA. Усовершенствован алгоритм OSTIA для обеспечения корректной трансляции цепочек, не входящих во входной язык. Проведён ряд экспериментов, в которых показано преимущество предложенного метода по сравнению с базовой версией OSTIA.

Ключевые слова:
Трансдуктор, конечный автомат, обучение, алгоритм, OSTIA, грамматическая индукция, множество негативных образцов.

Key words:
Transducer, finite automaton, learning, algorithm, OSTIA, grammatical inference, negative sample set.

Введение

В последние годы в связи с развитием человеко-машинных интерфейсов, основанных на технологиях распознавания рукописного ввода, речевого управления, а также невербального текстового ввода, работа по совершенствованию технологий разбора грамматических конструкций приобретает всё большее значение. Становится актуальным изучение особенностей языков, возникающих в различных социальных и технических системах, их свойств, таких как выразительность и избыточность, а также задача выполнения автоматического перевода из одного языка в другой.

Все задачи, перечисленные выше, решаются в рамках научной дисциплины, сформировавшейся в 60-х гг. XX в., – компьютерной лингвистики. В основе компьютерной лингвистики лежит утверждение о том, что передача информации и управление в технических системах может быть описано в терминах языков аналогично тому, как при помощи языков происходит общение между людьми. Компьютерная лингвистика предоставляет средства для анализа и синтеза языков, как строго формализованных, используемых в технических системах, так и естественных.

Выполнение перевода на конечных машинах

Основным инструментом компьютерной лингвистики являются конечные машины, представляющие собой варианты абстрактных машин Тьюринга: конечные автоматы для распознавания и конечные трансдукторы (двухленточные автоматы) для перевода языков. Именно трансдукторы и выполнение переводов с их помощью являются объектом данного исследования.

Будем называть переводом с языка L на язык L' любое отображение ϕ: $L \to L'$. В случае языков такое отображение может быть эффективно описано в терминах конечных машин состояний. Для выполнения перевода используется трансдуктор (двухленточный автомат) – машина состояний с двумя лентами, одна из которых содержит входную цепочку $v \in L$, а на второй в процессе работы машины формируется выходная цепочка, $v' \in L'$ такая, что $v' = \phi(v)$.

Вывод трансдуктора формируется на основании внутренней функции ψ: $Q \times \Sigma \times Q \to \Gamma^*$, которая для каждого перехода ($q \in Q$, $a \in \Sigma$, $q' \in Q$) задаёт цепочку из Γ^*, которая будет записана на выходную ленту при прохождении трансдуктором данного перехода. На рис. 1 представлен пример трансдуктора с двумя состояниями q_0 и q_1. Значения функции ψ указаны над переходами после двоеточия. Символ λ обозначает пустую строку.

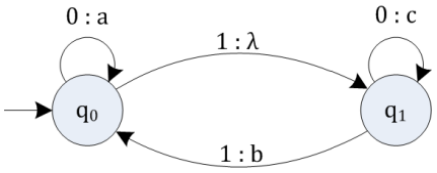

Рис. 1. *Пример трансдуктора*

Задача обучения конечных трансдукторов

Актуальность задачи обучения конечных машин, в частности трансдукторов, объясняется несколькими причинами. *Во-первых*, если говорить о задаче перевода естественных языков, то такие языки, как правило, слишком сложны и разнообразны, чтобы их можно было изначально описать в полной конечной форме, а значит, необходим механизм изучения новых пар выражений. *Во-вторых*, применительно к искусственным языкам, которые используются в технических системах, и, как правило, хорошо специфицированы и значительно более просты, возможны ситуации, когда спецификация языка точно неизвестна (например, секретна) или отсутствует.

Сформулируем задачу обучения конечных трансдукторов. Пусть Σ и Γ – алфавиты, $L \subseteq \Sigma^*$ и $L' \subseteq \Gamma^*$ – языки над этими алфавитами, а ϕ: $L \to L'$ – перевод с языка L на язык L'. Тогда задачу обучения трансдуктора можно сформулировать следующим образом: дано (конечное) множество

$S^+ \equiv \{(v_i, \phi(v_i)) | v_i \in L,\ i=1..n\}$ пар цепочек, найти трансдуктор T такой, что $\forall v \in L \Rightarrow T[v]=\phi(v)$. Здесь $T[v]$ обозначает цепочку на выходной ленте трансдуктора T после завершения работы с входом v.

Нужно сделать два важных замечания. *Во-первых*, задача грамматической индукции трансдуктора (ЗГИТ) в общем случае не имеет единственного решения, поскольку трансдуктор, задающий каждый конкретный перевод двух языков, как правило, не единственен. *Во-вторых*, указанная формулировка задачи не позволяет в общем случае установить правильность решения, поскольку требует знания искомого перевода $\phi(v)$.

ЗГИТ во многом схожа с аналогичной задачей для конечных автоматов, однако в случае трансдукторов задача усложняется тем, что соответствующий алгоритм должен изучать одновременно два языка, а также отношение между ними, что повышает не только трудоемкость алгоритма, но и вносит дополнительную неопределённость в процесс индукции.

Алгоритм OSTIA для обучения трансдукторов

В 1993 г. Хосе Ончина (Jose Oncina) и его коллеги предложили алгоритм OSTIA (Onward Subsequential Transducer Inference Algorithm), который стал первым алгоритмом, позволившим решать ЗГИТ [2]. Алгоритм OSTIA основан на принципах, ранее показавших свою эффективность в алгоритме RPNI [1], который успешно применяется для обучения конечных автоматов. В качестве исходных данных алгоритм OSTIA принимает множество S^+ пар, задающих перевод отдельных цепочек входного языка в соответствующие им цепочки выходного языка.

На основании входного множества пар алгоритм строит так называемый префиксный трансдуктор T_0 (Prefix-tree Transducer, PTT), который имеет вид дерева и является первым приближением на пути к искомому трансдуктору. Префиксный трансдуктор выполняет перевод в точности множества цепочек, заданных в обучающем множестве пар, т. е. $\forall (v,v') \in S^+ \Leftrightarrow T_0[v]=v'$. Пример префиксного дерева для множества из пяти пар, полученных для трансдуктора на рис. 1, приведён на рис. 2.

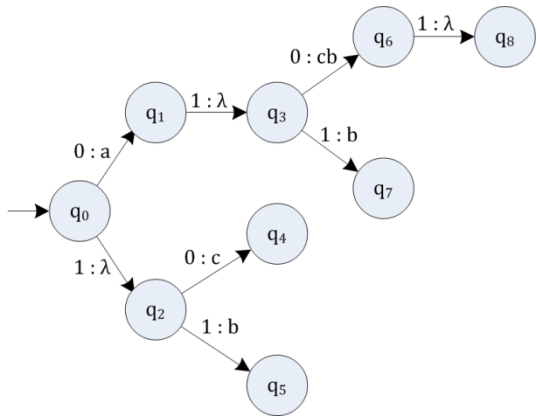

Рис. 2. *Пример префиксного дерева для трансдуктора*

При добавлении пары (v, v') из S^+ в трансдуктор сначала находится ветвь состояний, соответствующая входной цепочке v пары (если такая ветвь полностью или частично отсутствует, недостающие состояния добавляются в трансдуктор). После того, как нужная ветвь найдена, вывод последнего состояния цепочки устанавливается равным v'.

Таким образом, прочитав соответствующую входную цепочку, трансдуктор пройдёт по всей ветви и, остановившись в последнем её состоянии, выполнит вывод на ленту соответствующей выходной цепочки. Так как не все цепочки, появляющиеся в трансдукторе, присутствуют во входном множестве пар, то состояниям, вывод которых не определён, присваивается специальный символ $\perp$.

Поскольку в трансдукторе выводы распределены по всем его переходам, то вторым шагом алгоритм смещает выводы от крайних состояний максимально близко к корню дерева. Полученный трансдуктор носит название Onward Subsequential Transducer. Смещение выводов основано на следующем наблюдении: если некоторый выходной префикс является общим для всех переходов, выходящих из данного состояния, то его можно перенести в качестве суффикса на все переходы, входящие в это состояние, не изменив перевода, выполняемого трансдуктором.

Затем алгоритм переходит к индукционной фазе, которая решает одновременно две задачи: *во-первых*, минимизирует трансдуктор путем отыскания и слияния одинаковых поддеревьев и, *во-вторых*, путём индукции дополняет перевод новыми парами, восстанавливая отсутствующие цепочки и циклы на основании цепочек и циклов, уже присутствующих в трансдукторе.

Первичное восстановление трансдуктора из множества пар-образцов приводит к тому, что одно и то же состояние исходного трансдуктора отображается в одно или несколько состояний префиксного дерева. Таким образом, задача полного восстановления исходного трансдуктора в индукционной фазе состоит в «сворачивании» префиксного дерева к исходной форме путём отыскания и объединения копий состояний.

В базовой версии OSTIA единственным критерием объединения состояний является одновременная совместимость их выводов и поддеревьев. Два поддерева трансдуктора считаются совместимыми, если для любой цепочки v, которая может быть прочитана в обоих поддеревьях, цепочки, генерируемые этими поддеревьями на v, совместимы. Цепочки ω и ω' считаются совместимыми, если предикат *compat* $(\omega \in \Gamma^* \cup \{\perp\}, \omega' \in \Gamma^* \cup \{\perp\}) \equiv \omega=\omega' \vee \omega=\vee \perp \omega'=\perp$ истинен.

Данное условие гарантирует, что при слиянии состояний поддерево объединенного состояния останется детерминированным. Любая найденная пара совместимых состояний сразу объединяется, а их поддеревья сливаются. Процесс поиска и слияния пар продолжается до тех пор, пока не останется пар состояний для объединения.

Недостатки алгоритма OSTIA

Несмотря на то, что само появление алгоритма OSTIA является значительным прорывом в решении ЗГИТ, этот алгоритм обладает и рядом недостатков. Первый и наиболее очевидный из них проистекает из «жадной» природы алгоритма. В процессе выполнения объединений состояний неизбежно возникают ситуации, когда возможно выполнение двух и более (возможно несовместимых) объединений. Для того, чтобы детерминировать процесс обучения OSTIA проверяет состояния и выполняет объединения строго в порядке возрастания длин префиксов состояний. Такой порядок гарантирует детерминированную работу алгоритма, но не гарантирует принятия на каждом шаге оптимальной пары для слияния.

При этом объединение состояний, выполненное на очередном этапе алгоритма, может в будущем привести к невозможности выполнения других объединений. Данная проблема может быть решена путём введения процедуры сравнения (ранжирующей функции) доступных на каждом этапе для объединения пар состояний с точки зрения некоторых критериев оптимальности такого объединения.

Вторым существенным недостатком алгоритма является то, что он не позволяет производить вероятностную оценку правильности перевода, выполненного на обученном трансдукторе. Процесс индукции это процесс построения выводов на основании исходных данных и выводов, сделанных ранее, а каждое логическое заключение характеризуется некоторой вероятностью. В OSTIA, тем не менее, неявно предполагается, что переводы, выведенные в процессе обучения, так же достоверно корректны, как и те, которые были заданы в исходном обучающем множестве.

Единственным входом алгоритма OSTIA является множество пар, однако это не вся информация, которая может быть использована при решении ЗГИТ. В современных алгоритмах обучения конечных автоматов помимо множества цепочек, входящих в язык – положительного множества, используется также множество цепочек, заведомо не являющихся корректными цепочками языка. Это множество, называемое негативным, позволяет в алгоритмах обучения автоматов повысить вероятность достижения правильного решения.

Множество негативных образцов может быть также использовано и в алгоритмах обучения трансдукторов, что позволит сдержать разрастание входного языка в процессе индукции, обеспечивая тем самым лучшее приближение к искомому трансдуктору.

Использование негативных образцов в алгоритме OSTIA

Рассмотрим возможность использования множества негативных образцов, т. е. множества $S^{-}\subseteq\Sigma^{*}/L$ в алгоритме OSTIA. Введение множества S^{-} требует новой формулировки ЗГИТ. Сформулируем ЗГИТ с множеством негативных образцов следующим образом: даны множества $S^{+}\equiv\{(v_i,\phi(v_i))|v_i\in L,i=1..n\}$ и $S^{-}\subseteq\Sigma^{*}/L$, найти трансдуктор T, такой что $\forall v\in L\Rightarrow T[v]=\phi(v)$ и $\forall v\in\Sigma^{*}/L\Rightarrow T[v]=\otimes$, где $\otimes$ – символ, соответствующий некорректному вводу.

Вариант использования множества негативных образцов, предлагаемый в данной работе, основан на анализе семантики негативного слова-образца. Можно говорить о том, что трансдуктор должен сопоставлять некорректным входным цепочкам символ $\otimes$ так же как и определённые выводы корректным входным цепочкам согласно множеству S^{+}.

При таком подходе множество S^{-} можно заменить множеством $S^{=}\equiv\{(v_i,\otimes)|v_i\in\Sigma^{*}/L,\ i=1..m\}$. В таком случае его можно объединить с множеством S^{+} так что $S=S^{+}\cup S^{=}$. Если подать множество S на вход алгоритма OSTIA, то префиксное дерево на первом шаге алгоритма будет построено так, что ветвям состояний, которые соответствуют некорректным цепочкам из S^{-}, будет присвоен вывод $\otimes$.

Далее необходимо обеспечить корректную обработку выводов состояний с символом $\otimes$ на последующих стадиях алгоритма. На втором шаге выводы состояний обрабатываются в операции смещения выводов к корню дерева. В данной ситуации важны два аспекта. *Во-первых*, поскольку символ $\otimes$ обозначает некорректный ввод, он должен быть ассоциирован с ветвью состояний, а значит её крайним состоянием, а не с отдельным переходом. Это означает, что вывод $\otimes$ не должен сдвигаться от состояния к переходу.

Во-вторых, важно, как будет интерпретирована цепочка, которая была сдвинута к состоянию с выводом $\otimes$ от других состояний. В данном случае вывод состояния используется в функции *lsp – longest common prefix* (самый длинный общий префикс), определённой на множестве всех подмножеств $\Gamma^{*}\cup\{\perp\}$. Функция *lsp* должна быть «прозрачна» для символа $\otimes$, а значит, её нужно доопределить на множестве всех подмножеств $\Gamma^{*}\cup\{\perp,\otimes\}$ следующим образом: $lsp\ (A\subseteq(\Gamma^{*}\cup\{\perp,\otimes\}))\equiv lsp\ (A/\{\perp,\otimes\})$.

В индуктивной фазе выводы состояний участвуют в тесте на совместимость, задаваемом предикатом *compat*. Этот предикат нужно также изменить, расширив область его определения до множества $(\Gamma^{*}\cup\{\perp,\otimes\})\times(\Gamma^{*}\cup\{\perp,\otimes\})$.

Таким образом, множество негативных образцов может быть без значительных затруднений введено в алгоритм OSTIA. Сравним результаты обучения трансдукторов при помощи базовой версии OSTIA и модифицированной версии алгоритма с использованием негативных образцов.

Ход и результаты экспериментов

В качестве задания при проведении экспериментов использовалась задача перевода римской записи чисел в эквивалентную арабскую запись, например CDVII → 407, идея которой была заимствована из работы [3]. Критерием качества обучения в данном эксперименте принят коэффициент

ошибки, рассчитанный как отношение числа неправильно переведённых образцов из тестового множества к размеру тестового множества.

Для формирования обучающих и тестовых множеств было сгенерировано множество из 10000 пар (римское число, арабский эквивалент). Затем пары из этого множества были нормально распределены между 50 фрагментами по 200 пар каждый. Также было сгенерировано (негативное) множество из 10000 цепочек, не являющихся корректными римскими записями чисел. Это множество было также разбито на 50 подмножеств по 200 цепочек.

Каждый эксперимент состоял из 50 раундов обучение-тест. Для базовой версии алгоритма в первом раунде в качестве обучающего множества использовался первый фрагмент, а оставшиеся 49 фрагментов составляли тестовое множество. Во втором раунде обучающее множество было составлено из первых двух фрагментов, а тестовое множество было образовано оставшимися 48 фрагментами и т. д. В последнем, 50-м, раунде все 50 фрагментов исходного множества использовались для обучения, таким образом, тестовое множество в этом раунде было пусто.

Модифицированной версии алгоритма на вход в каждом раунде подавалось обучающее множество, использованное для базовой версии, а также подмножество негативного множества такого же размера. Множество негативных образцов для каждого раунда формировалось аналогично обучающему множеству пар раунда. В первом раунде обучающее негативное множество было образовано первым фрагментом, во втором – первым и вторым и так далее. Тестовые множества соответственно включали все негативные образцы, которые не вошли в соответствующие обучающие множества.

Были проведены эксперименты трёх типов. В экспериментах первого типа сравнивалось, насколько эффективно способны оба алгоритма изучить непосредственно перевод с одного языка на другой. При этом в тестовое множество входили на каждом раунде только корректные римские числа, которые не использовались при обучении в данном раунде. Результаты этих экспериментов представлены на рис. 3, *а*.

Как видно из графиков на рис. 3, *а*, трансдуктор, обученный на комбинированном обучающем множестве, при равных размерах положительных обучающих множеств показывает больший коэффициент ошибки. Это объясняется тем, что алгоритм OSTIA в своей базовой версии, не имея ограничения в виде негативного обучающего множества, стремится делать неправильные выводы, которые приводят к принятию трансдуктором также и некорректных входных цепочек.

Эксперименты второго типа были направлены на то, чтобы показать, что положительное обучающее множество пар в отдельности не позволяет алгоритму OSTIA распознавать некорректные входные цепочки. В данном эксперименте в тестовое множество на каждом раунде входили только некорректные цепочки, которые не использовались при обучении на этом раунде. Результаты представлены на рис. 3, *б*.

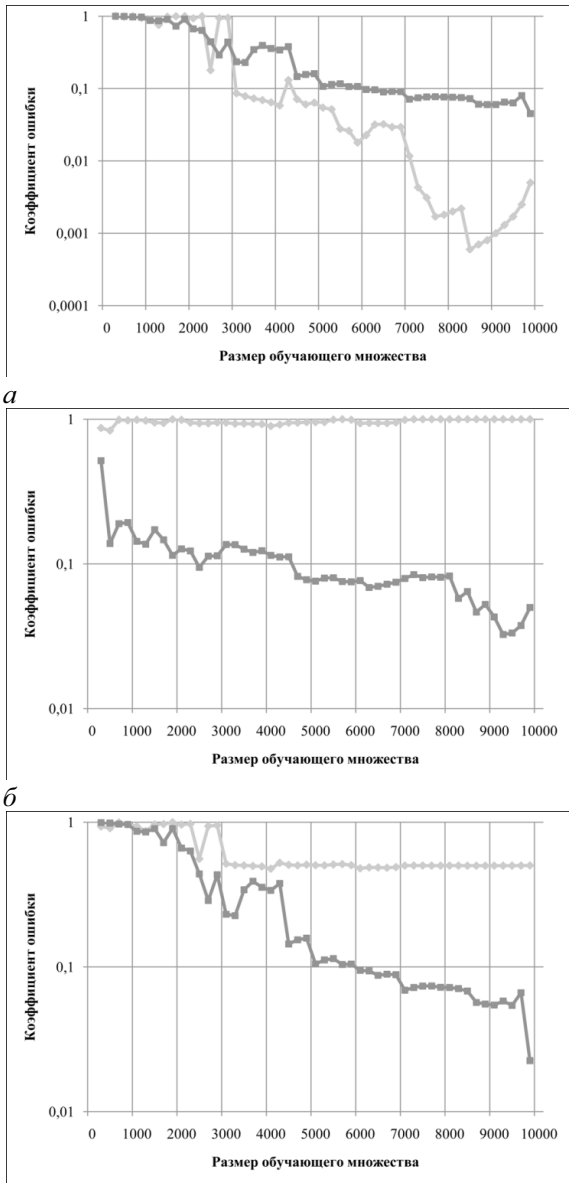

Рис. 3. *Результаты экспериментов на положительном (а), негативном (б) и комбинированном тестовых множествах: ♦ – трансдуктор, обученный на положительном множестве; ■ – трансдуктор, обученный на комбинированном множестве*

Графики на рис. 3, *б*, подтверждают выводы, сделанные на основании предыдущего эксперимента, о том, что в процессе индукции алгоритм OSTIA выводит переводы для произвольных, в том числе некорректных входных цепочек. Модифицированная версия алгоритма способна выделить общие закономерности некорректных входных цепочек. Можно заметить, что при равном размере обучающих множеств алгоритм OSTIA позволяет добиваться лучших результатов при изучении корректных входных

цепочек на основании положительного множества пар, чем при изучении некорректных цепочек на основании негативного множества. Это объясняется тем, что язык римских чисел конечен, за исключением возможно бесконечного префикса, а значит, принципиально более прост, чем бесконечный язык некорректных римских чисел.

В экспериментах третьего вида была рассмотрена ситуация, в которой на вход трансдуктора с равной вероятностью подаются как корректные, так и некорректные входные цепочки. Тестовое множество при этом на каждом раунде состояло из корректных и некорректных цепочек, не участвовавших в обучении на данном раунде и взятых в равном количестве. Результаты данного эксперимента представлены на рис. 3, *в*.

Как видно из графиков на рис. 3, *в*, на комбинированном тестовом множестве трансдуктор, обученный также на комбинированном множестве, обеспечивает значительно меньшую ошибку при равном размере обучающего множества. Более того, в данном эксперименте трансдуктор, обученный по базовому алгоритму OSTIA не смог преодолеть ошибки 0,50, в то время как модифицированная версия алгоритма достигла уровня ошибки 0,43 уже при размере обучающего множества 2400 пар.

Выводы

Предложен метод использования множества негативных образцов в алгоритме OSTIA, а также модифицированная версия алгоритма, использующая указанный метод. Метод обеспечивает в алгоритме OSTIA лучшее качество обучения в терминах коэффициента ошибки как функции размера обучающего множества на множестве всех входных цепочек по сравнению с базовой версией алгоритма. Введение негативного множества не изменяет временную сложность алгоритма как функцию размера входного множества, хотя размер самого входа увеличивается за счёт введения дополнительных фиктивных пар из негативного множества.

Показано, что в отсутствие информации о некорректных входных цепочках алгоритм OSTIA может выводить индуктивные заключения, которые не согласуются с изучаемым переводом, что приводит к трансдуктору с деформированным входным языком. В случаях, когда входной язык точно известен, и распознавание некорректных входных цепочек не требуется, базовая версия алгоритма является более простым и эффективным решением задачи грамматической индукции для трансдукторов.

Исследования поддержаны грантом РФФИ № 11–07–00027-а.

СПИСОК ЛИТЕРАТУРЫ

1. De la Higuera C. Grammatical Inference. – N.Y.: Cambridge University Press, 2010. – 417 p.
2. Oncina J., Garcia P., Vidal E. Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1993. – № 15. – P. 448–458.
3. Oncina J. The Data-Driven Approach Applied to the OSTIA Algorithm // Proceedings of Icgi'98, July 12–14, 1998. – N.Y.: Springer, 1998. – P. 50–56.

Поступила 10.02.2011 г.