Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Модель и алгоритм улучшения распознавания частей речи в текстах, содержащих ошибки

Курсовая работа студента 445 группы

Ткаченко Максима Владиславовича

Научный руководитель: Л.С. Выговский,

ассистент кафедры АСОИУ, СПбГЭТУ ЛЭТИ

Санкт-Петербург

2010

Оглавление

1

Введение

Научить машину понимать естественный язык - довольно привлекательная задача. Первые системы, работавшие с ограниченным набором словарного запаса, показывали хорошие результаты при обработке выражений простого человеческого языка. Однако, когда эти системы столкнулись со сложностью и неоднозначностью окружающего мира, то их создателей постигло разочарование. Распознавание выражений естественного языка требует огромных знаний об окружающем мире, от такого рода системы требуется взаимодействие с окружающей действительностью, выявление его закономерностей, что на текущем этапе развития искусственного интеллекта кажется затруднительным.

Тем не менее, обработка текстов на естественном языке не такое бесполезное занятие. Машина способна обрабатывать гораздо больший объем информации, по сравнению с человеком. На рынке программного обеспечения (главным образом западного) существует множество так называемых приложений "добычи текста"(Text Mining), которые извлекают высококачественную информацию из массивов текстов, написанных на привычном для человека языке. Приложения довольно успешно решают задачи классификации и кластеризации текстов, автореферирования и т.д. Text mining приложения неотрывно связаны с концепциями информатики и вычислительной лингвистики, т.е. с областью обработки естественных текстов (Natural Processing Language, или более коротко NLP).

Данная работа в большей степени рассматривает морфологический аспект автоматической обработки текстов, задачу распознавания частей речи (part-of-speech tagging). Хотя, морфоанализ представляется собой довольно традиционную область автоматической обработки текстов, все время появляются новые лингвистические процессоры, в то время как некоторые старые отмирают или же, напротив, дорабатываются и развиваются с учетом новых задач. Отсутствует универсальный текстовый процессор, который решал бы все необходимые задачи. В результате имеется рынок компьютерных реше-

ний, отличающихся функциональными возможностями, детальностью, форматами и т.д.

Рассмотрим вопрос определения частей речи более подробно. Нам дан текст, написанный на русском языке, для каждого слова в этом тексте хочется определенно указать часть речи, к которой оно принадлежит (здесь термин “слово” употребляется как синоним термина “словоформа”, обозначающий конкретное слово в конкретной грамматической форме). Данная деятельность сопряжена с некоторым набором проблем. Рассмотрим одну словоформу без какого-либо контекста, пусть это будет слово “печь”. Что это, глагол или существительное? - Без точного указания контекста нельзя сказать наверняка. Аналогичная ситуация происходит со словом “три”. Данное явление называется омонимией.

Язык отнюдь не статичная структура, он достаточно бурно развивается, и порой непонятные сочетания букв могут приобретать конкретный смысл в той или иной группе людей, а соответственно и определенную часть речи. Для идентификации таких слов требуется предсказание части речи. В прозе Набокова, Мандельштама, Гоголя и т.д. часто встречаются неологизмы описание которых нельзя встретить в морфологических словарях [9]. Например, “... падая, хрустя, хохоча с **запышкой**, влез на сугроб...”.

Подобного предсказания требуют слова, которые пришли к нам из других языков, однако их анализ сопряжен с дополнительной сложностью: такие слова как правило не подчиняются законам словообразования русского языка.

Требуется также отличать имена собственные от слов одинакового написания. “Станция метро Пролетарская” - это не тоже самое, что и “пролетарская поэзия”. В принципе такого рода неоднозначность подходит к определению омонимии, однако лучше вынести эту проблему в отдельный пункт, ибо сам по себе вопрос распознавания имен собственных в тексте является предметом активных исследований.

Как правило, тексты пишут люди, и ожидать от них, что все предложению будут написаны правильно в соответствии с канонами русского языка, - довольно оптимистично. В большей степени это касается живой переписки: чаты, блоги и т.д. Случайно пропущенная буква может стать серьезной проблемой при проведении морфоанализа. Тексты, в которых в большом количестве присутствуют грамматические, синтаксические ошибки, выделяют в отдельную группу, так называемых, грязных текстов. Помимо переписки, это могут быть еще и автоматически распознанные сканы или речь.

Данная работа нацелена на построение модели и алгоритма, которые позволили бы улучшить распознавание частей речи в текстах с ошибками. Следует отметить, что выражение “частей речи” здесь понимается в более широком смысле, нежели просто су-

ществительное, прилагательное и т.п. Разметка представленная в данной работе может быть легко расширена, например, чтобы учитывать некоторые грамматические характеристики, такие как число у имен существительных...

Снятие морфологической неоднозначности, полезно во многих приложениях компьютерной лингвистики. В частности, такого рода морфологический анализатор, может быть использован для последующего синтаксического и семантического анализа текста. В системах информационного поиска выделение частей речи может способствовать повышению точности некоторых классов поисковых запросов и сократить объем хранимой информации в индексе.

2

Обзор существующих методов

Большинство существующих методов описывают решение проблем, связанных со снятием омонимии, предсказанием частей речи, выделением имен собственных раздельно, каждая из представленных задач имеет свою практическую значимость, однако для данной работы необходима некоторая комбинация представленных методов. Полученная алгоритмическая модель должна эффективно справляться с поставленной задачей, причем с довольно хорошей точностью.

После появления “Грамматического словаря” Зализняка [6, 9] базовые методы выделения морфологической информации основаны просто на подглядывании в словарь. Этот подход хорош, когда словоформа содержит всего одну морфологическую интерпретацию, если интерпретаций много, то приложение должно уметь оценивать контекст слова, с целью выявления корректных грамматических характеристик. Заметим, что на обслуживание словаря уходит довольно много времени, и вряд ли даже самый большой словарь будет содержать сленговые конструкции, используемые определенной группой людей, то есть словарь в какой-то мере зависит от предметной области. Предсказание частей речи так же в большинстве своем основано на словарном подходе. Главная идея здесь - провести аналогию со словами, для которых уже известны морфологические характеристики характеристики. Так как в русском языке большую часть грамматических свойств несет в себе окончание слова, то имеет смысл сравнивать последние буквы слова неизвестной словоформы с окончаниями слов, содержащихся в словарях. Такой подход описан группой исследователей Aot.ru [5] (автоматическая обработка текстов). Однако, данный алгоритм может дать слишком много лишних грамматических интерпретаций, и здесь уже остро встает вопрос снятии омонимии, причем с очень большим количеством альтернатив. Конечно, в данном случае можно рассматривать лишь часто используемые грамматические свойства, характерные данному окончанию слова, но к в общем случае такой подход будет давать неверный результат.

Существующие подходы к снятию омонимии традиционно основываются на синтаксическом разборе либо на некоторых статистических методах. Рассмотрим подробнее синтаксический подход.

Пусть нам известны правила, по которым мы можем связывать слова в предложениях, причем для разных грамматических форм эти правила, теоретически, разнятся. Тогда корректно разобрав предложение, можно делать выводы о морфологии слов, составляющих предложение. Такой подход связан с довольно естественной проблемой: сам по себе синтаксический анализ сложная задача. Казалось бы для корректного синтаксического анализа текста необходимо сначала однозначно определить морфологические характеристики каждого слова. Кроме того, обработка предложения со большим изначальным числом связей ведет к значительному замедлению скорости анализа. Тем не менее большая доля морфологических процессоров использует именно синтаксический подход к разрешению омонимии. В качестве примера таких систем можно привести: лингвистический процессор ЭТАП и синтаксический анализатор Диалинг.

Статистические методы для разрешения морфологической омонимии применительно к русскому языку стали использовать сравнительно недавно. Зеленков и др. [7] предложили алгоритм, предназначенный для разрешения морфологической омонимии у слов, совпадающих лишь в нескольких грамматических формах. Метод основывается на использовании автоматически полученного словаря контекстов, выведенного из уже размеченных текстов. Предложенная алгоритмическая модель достаточно эффективна, однако ее применение для разбора текста с ошибками может быть затруднительно, ибо это потребует наличия словаря контекстов, содержащего ошибки, что не совсем корректно.

Известным вероятностным подходом для решения задачи разрешения омонимии, является алгоритм, основанный на использовании скрытой Марковской модели (Hidden Markov Model tagging). Метод требует предварительного обучения системы на уже размеченной выборке текстов, а учитывая богатство русского словообразования и словоизменения для этого требуется размеченные корпуса очень большого объема. Точность работы алгоритма для русского языка составляет 97.26% [4]. Использование некоторой модификации данного подхода не лишено смысла, однако сам алгоритм связан с трудоемкими вычислениями (связанные с алгоритм Витерби), что плохо сказывается на скорости обработки текстов.

Обратимся к опыту морфологического разбора английского языка [2], как наиболее исследованного. Хотя словообразование русского языка считается более богатым, некоторые модели и алгоритмы морфологического анализа английских текстов могут быть использованы и в русском. Большую часть методов анализа английского языка состав-

ляют статистические подходы, основанные на идеях машинного обучения, в качестве примера можно привести уже рассмотренный метод скрытой Марковской модели, метод опорных векторов (Supprot Vector Machines, более кратко - SVM), метод максимальной энтропии (Maximum Entropy), деревья решений (Decision Trees). Среди перечисленных методов стоит выделить Supprot Vector Machines, точность которого составила 97.2% при тестировании на текстах новостных статей из корпуса The Wall Street Journal, что является хорошим результатом.

Основная идея метода опорных векторов — поиск разделяющей гиперплоскости с максимальным зазором между векторами двух различных классов. То есть, если должным образом представить входной текст в виде векторов, то можно проводить классификацию каждого слова по классам соответствующим частям речи. Естественно, для для нахождения разделяющей гиперплоскости потребуется уже размеченный набор текстов, но с учетом специфики метода он требует не столько количества примеров, а сколько их показательности. Механизм метода опорных векторов довольно прост, но эффективен как показывает практика, а гибкость алгоритма позволяет успешно сочетать его с уже существующими методами определения частей речи и снятия омонимии.

Чтобы можно было использовать методы машинного обучения для анализа текстов, необходимо уметь этот текст представлять в удобном для машины виде, в частности следует знать какие характеристики следует включать в понятное для машины представление. К сожалению, автору известны мало работ по применению статистических методов к морфологическому анализу русского языка, поэтому данный обзор будет ограничен лишь общими замечаниями. Для того чтобы определить часть речи выбранного слова, обычно в качестве характеристик классификации берется некоторый его контекст, то есть машина знает информацию о соседних словоформах, их окончание, часть речи, возможную часть речи (если она еще не может быть точно определена), некоторые характеристики написания слова. Например, для выделения имен собственных в тексте, часто смотрят на то, написана ли первая буква прописью, стоит ли слово в начале предложения, содержит ли оно латинские буквы, цифры.

Хотя в области статистического морфологического анализа есть некоторые результаты, многое еще не исследовано. Не в полную силу использованы уже предложенные подходы, хотя, как можно заметить, и их помощью достигаются неплохие результаты. Некоторые методы можно успешно совместить, что теоретически позволило бы создать достаточно эффективный морфологический процессор.

Можно заметить, что интерес к морфологическому анализу в большинстве своем поддерживается со стороны компаний занимающихся информационным поиском. Мно-

гие из представленных работ выполнен при поддержке компании Yandex [4, 7]. Однако с развитием рынка приложений интеллектуального анализа текстов, этот список можно будет увеличить.

В данной работе построена модель текста, удобная для работы со статистическими подходами. В качестве алгоритмической части использован метод опорных векторов в сочетании со словарем Зализняка.

3

Основная часть

3.1 Построение модели словаря Зализняка

Было бы глупо отбрасывать опыт словарных методов в определении морфологии слов, тем более, если словоформе можно приписать только одну разумную часть речи. Например, словоформа “кошка” имеет вполне определенную морфологию: существительное женского рода, именительного падежа, единственного числа. Этому слову сложно придумать альтернативы, это определенно существительное, а не глагол и не прилагательное. Разумно прекратить анализ, если для слова есть некоторая однозначная морфологическая информация. Для этого требуется построить модель некоторого словаря, и эффективный алгоритм возвращающий возможные морфологические альтернативы для входной словоформы. Эффективность обусловлена тем, что подглядывание в словарь не должно влиять на скорость работы последующего анализа, для более точной оценки стоимость работы алгоритма разрешения неоднозначность. Если для словоформы возможно несколько морфологических альтернатив, то анализ следует продолжить до снятия омонимии. Если в словаре не существует статьи соответствующей входному слову, происходит процесс предсказания части речи.

Построенная модель базируется на русском морфологическом словаре Диалинг, исследовательской группы Aot.ru, в основу которого входит грамматический словарь А.А.Зализняка. Зализняк в своих работах наиболее полно описал русское словообразование и грамматику, что позволило разрешить внушительную часть проблем, связанных с автоматическим определением характеристик слова. Словарь насчитывает около 161 тыс. лемм (лемма - каноническая форма лексемы, лексема - слово, рассматриваемое со всей совокупностью своих форм), что дает в итоге морфологическую информацию для 3,5 миллионов словоформ.

Изначально для работы со словарем была выбрана структура основанная на Radix

Tree, с небольшими дополнениями. Построенная структура представляет из себя дерево с корнем в виде пустого символа. Остальные вершины представляют из себя различные части слов, ребра от корня к листьям связывают эти части в словоформы (то есть если пройти от корня дерева до какого-нибудь листа и выписывать последовательно все содержимое вершин, по которым проходит путь в графе, выписанные буквы составят словоформу, либо несколько словоформ включенных друг в друга). Каждое слово представленное в дереве оканчивается специальной вершиной, у которой есть ссылка на его морфологию. Таким образом мы получаем структуру, которая работает за линейное от длинны входа время при запросе грамматических характеристик. Однако данная модель обладает существенным недостатком - это память. Учитывая то, что дерево строиться из огромного набора различных словоформ, а русский язык имеет богатое словообразование, построенная структура обладает сильном ветвлением - более чем гигабайт памяти требуется, чтобы работать с таким словарем. Для того чтобы решить проблему памяти, автором был предложен алгоритм преобразующий построенное дерево в более компактный граф. Итоговая структура позволила сократить требование к ресурсам памяти на 99%. Рассмотрим алгоритм преобразования более подробно.

Заметим несколько особенностей русского языка, он флективен (то есть большую морфологическую нагрузку несет в себе конец слова слова), и в тоже время окончания часто повторяются. Таким образом можно рассчитывать на то, что некоторые поддеревья близкие к листьям будут повторяться, и можно будет избавиться от ненужных дубликатов. Применяя предложенные преобразования над исходном деревом, мы потеряем возможность добавлять в него новые элементы, но в данном случае эту жертву можно принести, так как динамика изменения словаря не такая большая.

Для каждого поддерева в исходном дереве просчитывается его хэш-значение, затем в дереве происходит поиск дубликатов, естественно, проверять на идентичность стоит только те поддеревья, у которых совпадают хэш-значения. Дубликаты удаляются из памяти.

Полученный после преобразования словарь довольно компактен. В итоге дерево было закодировано в целочисленный массив который составил около 5 мегабайт.

Построенный словарь может вполне быть использован как отдельный результат работы, и в последствии будет активно использован для построения алгоритмической модели.

3.2 Построение алгоритма морфологического анализа

3.2.1 Введение в метод опорных векторов

Метод опорных векторов относительно новый метод, используемый для бинарной классификации входных данных. Его основная идея заключается в том, чтобы найти гиперплоскость, которая разделит вектора из пространства размерности d в точности на два класса [1]. Размерность пространства определяется количеством наблюдаемых характеристик.

Рассмотрим математические аспекты метода, которые помогут лучше понять применимость и сложность метода. Любую плоскость из пространства R^n можно представить в виде:

$$w \cdot x + b = 0.$$

Здесь w - вектор ортогональный плоскости, b - коэффициент смещения плоскости. Если будут найдены параметры w и b искомой гиперплоскости, то процесс классификации вектора x в общем случае представляет из себя простое вычисление формулы, хотя и довольно трудоемкое, при большой размерности пространства,

$$y = sign(w \cdot x + b)$$

y - принимает значение 1 либо -1, если входной вектор относится к тому или иному классу соответственно. Интуитивно понятно, что надо выбирать гиперплоскость, которая “хорошо” отделяет два множества. Это “хорошо” значит, что расстояние от множества точек каждого из классов до гиперплоскости максимизируется.

Заметим, что параметры $\{w, b\}$ и $\{\lambda w, \lambda b\}$ задают одну и ту же плоскость, поэтому удобнее выбирать некоторую каноническую гиперплоскость. Назовем таковой плоскость, отделяющую тренировочный набор, зазором в единицу (далее работаем только с каноническими параметрами). Тренировочное множество - это набор n векторов используемых для поиска разделяющей гиперплоскости, для каждого из такого вектора известен его корректный класс, (x_i, y_i) $i = 1..n$, где x_i - вектор из пространства R^n, y_i - указатель класса ($y_i \in \{1, -1\}$). Учитывая данные определения, можно строго записать некоторые свойства искомой плоскости:

$$x_i \cdot w + b \geq +1$$

при $y_i = +1$

$$x_i \cdot w + b \leq -1$$

при $y_i = +1$

Или более компактно:

$$y_i(x_i \cdot w + b) \geq 1$$

Рассмотрим расстояние от искомой гиперплоскости, до множества точек из тренировочного набора:

$$d(\{w, b\}, x_i) = \frac{y_i(x_i \cdot w + b)}{||w||} \geq \frac{1}{||w||}$$

Из неравенства видно, чтобы найти нужную гиперплоскость надо уменьшать длину вектора w, (под оператором $||.||$ подразумевается норма вектора, те его длинна), тем самым увеличивая зазор между тренировочным множеством и гиперплоскостью.

Рассматриваемая задача поиска разделителя сводиться к задаче квадратичного программирования, что позволяет решать задачу уже известными алгоритмами оптимизации. В этом факте можно убедиться самим заглянув в статьи В.Вапника [3].

К сожалению данные не всегда точно отделимы, в линейном случае, и если не совершать никаких преобразований входных данных, гиперплоскость можно построить лишь приближенно, сразу получая некоторые ошибки классификации. Однако и в данном случае можно добиться успешного решения задачи в целом. Существуют методы, позволяющие преобразовывать входные данные в пространства других размерностей, чтобы увеличить их свойства отделимости, но в данной работе они не рассматриваются ввиду двух причин: этап преобразования сильно замедляет работу алгоритма как обучения так и классификации, и на практике изменение входных данных дало лишь незначительное улучшение результатов.

Для решения поставленной задачи требуется классификация слов более чем по двум классам, а именно по 14-ти классам представляющим соответствующие части речи. Для этого в работе используется так называемая классификация по типу “один против всех”. Для каждой части речи тренируется отдельный классификатор, отделяющий слова данной части речи от слов всех других частей речи. Такой тип обучения дает дополнительные бонусы: для каждого слова можно использовать только те классификаторы, которые ему действительно необходимы. Если происходит процесс снятия омонимии, то это лишь некоторые из них, если процесс предсказания речи используется все натренированные модели.

Предположим, что классификаторы сработали не совсем точно - для некоторого слова было определено две части речи. Чтобы разрешить такие противоречия выбирается класс который расположен наиболее удаленно, от разделяющей плоскости, то есть имеющий наибольшее значение $||w \cdot x + b||$.

В некоторых случаях классификатор не может точно определить часть речи входного слова, то есть все классификаторы сработали на класс “против всех”. Чтобы разобраться с этой ситуацией выбирается часть речи, у которой классифицирующая гиперплоскость расположена как можно ближе к характеристическому вектору. Таким образом минимизируется соответствующая величина $||w \cdot x + b||$

В качестве тренировки классификаторов была выбрана сторонняя библиотека уже с успехом используемая исследователями для проведения опытов - lightsvm. Для обучения необходимо подготовить входной текст к заданному векторному формату. На выходе приложение возвращает параметры и коэффициенты необходимые для вычисления гиперплоскости. Ввиду того, что алгоритм совмещает работу словаря с работой классификатора, приложение классификации дополнительно было переписано на языка Java (язык на котором был реализован словарь). Подведем некоторые итоги, для каждой части речи тренируется классификатор основанный на методе опорных векторов. Расширение бинарной классификации идет по методу “один против всех”. Полученный классификатор используется в алгоритме совместно со словарем Зализняка, что позволяет улучшить точность выполнения алгоритма и увеличить его скорость работы.

3.2.2 Построение модели текста

Чтобы алгоритм работал для хорошо и на текстах содержащих ошибки и нет, необходимо выбрать достаточно универсальную модель представления текста. Для начала отметим, что на вход SVM-классификатору поступает вектор пространства R^n, где в качестве n выступает количество свойств слова, которые мы хотим рассматривать. Таким образом, в качестве характеристик следует брать достаточно показательные элементы, чтобы лишний раз не замедлять работу алгоритма. С целью дополнительно ускорения работы метода, для построенного классификатора в качестве характеристик слова берется вектор состоящий лишь из единиц и нулей, таким образом при подстановке вектора в уравнение гиперплоскости, мы избавляемся от довольно медленной операции умножения. Вернемся к выбору характеристик. Перечислим свойства, которые уже использовались при классификации текста с целью определения части речи (для английского языка [2]):

- Словоформа. Здесь подразумевается, что у нас есть некоторое множество слов, которые, по нашему мнению, влияют на работу классификатора, пусть размерность данного множества есть L. Каждому слову припишем свой номер от 1 до L. Тогда, если соответствующее слово содержится в множестве, причем порядковый номер словоформы есть i, то в качестве характеристического вектора берется вектор длинны l, состоящий из нулей, кроме i-ого элемента, который является единицей. Такой метод получения характеристики будем называть словарным.

- Контекст. Обычно рассматривается некоторый контекст слова, так называемое “окно”, в него входит слово подлежащее классификации, а также определяются характеристики некоторого количества соседних слов, которые, вероятно, могут повлиять на результаты.

- Части речи. В качестве характеристик берутся части речи для слов контекста, если они могут быть определены, возможные части речи для классифицируемого слова, и для слов которые в последующем будут проанализированы.

Для английского языка этих свойств оказалось достаточно, чтобы показать 97% точности. Однако сохранение такой модели для русского языка дало результаты лишь немногим отличающиеся от простого словарного подхода.

Начнем модифицировать модель, с целью выявить характеристики, позволяющие проводить хорошую классификацию текста и без ошибок.

Начнем с контекста слова. Для простоты и универсальности метода контекст слова берется только из одного предложения. Далее из него выбираются некоторые сосед-

ние словоформы к данной. По утверждению Зеленкова, в русском языке достаточно учитывать только 5 соседних слов (3 до и 2 после) при определении корректных грамматических характеристик. Поэтому в модель входят характеристики только этих пяти слов.

Из характеристик классифицируемой словоформы выбираются следующие:

- Написано ли слово с прописной или строчной буквы.
- Содержит ли числа.
- Содержит ли символы иностранных языков.
- Находится ли в начале предложения.

Данные характеристики выбраны с целью определить имена собственные, числительные, состоящие из цифр, которые обычно не содержаться в словаре, и слова иностранного языка.

Определим характеристики более общие, взятые для всех слов контекста. Для каждого слова определяется его лемма и используется как словарная характеристика. В качестве словаря используется список наиболее употребляемых лемм, который был автоматически получен при анализе текстов новостных статей интернет сайтов. Для получения леммы исходного слова использовался набор эвристических алгоритмов, открытого проекта Snowball. Чтобы сделать модель еще более устойчивой к грамматическим ошибкам совершенным в середине слова, в добавок к лемме берутся первые 5 букв словоформы (либо меньше, если того требует необходимость).

С учетом флективности русского языка, в качестве характеристик необходимо использовать окончания слов. Из тренировочного набора было сгенерировано 3 словаря, однобуквенных, двубуквенных и требуквенных концовок слов. Использование сразу трех типов окончаний опять же позволяет сделать алгоритм более гибким в случае, если ошибки были допущены в конце слова слова.

Как и в английском языке, предположения о части речи текущего слова лучше проводить со знанием частей речи предыдущих слов контекста. Для классифицируемого и последующих слов известны их возможные морфологические характеристики, которые могут быть получены путем подглядывания в словарь. Следует отметить, что хотя автор рассматривает некоторую контекстную обработку слова, в качестве характеристик не следует брать знаки препинания в тексте, потому что как раз пунктуация чаще всего страдает при переписке в чатах, при сканирование из распознавании текстов маленькую точку очень легко потерять.

В процессе тестирования алгоритма были опробованы также некоторые другие свойства, такие как

- длинна слова;
- содержит ли слово знаки пунктуации;
- написано ли оно полностью прописью.

Однако, использование этих характеристик только ухудшало точность процесса. Чтобы провести классификацию и обучение необходимо, соответственно, вцепить из текста необходимые характеристики, привести их в понятный для классификатора вектор и подать на вход алгоритму.

3.2.3 Тренировка классификатора и тестирование

Для тренировка и тестирования классификатора необходим некоторый довольно объемный набор текстов. В качестве такого набора было взято подмножество Корпуса Национального Языка (RusCorpora) [8]. Национальный Корпус - это ресурс содержащий предложения вручную аннотированные синтаксической, семантической, морфологической информацией. Тексты Корпуса предоставляются для исследований в области лингвистики. RusCorpora имеет свою довольно специфическую разметку, которая не совпадает с разметкой морфологического словаря от группы Диалинг. Во избежание неоднозначности словарные статьи были приведены в согласование с разметкой Корпуса Национального Языка которая впоследствии и стала основной для алгоритма. Приведем размету используемую для частей речи в RusCorpora:

- S — Существительное: завод, я
- A — Прилагательное: новый, мой, второй
- V — Глагол: работать, нравиться
- ADV — Наречие: плохо, отчасти
- NUM — Числительное: пять, 2
- PR — Предлог: в, между, вопреки
- CONJ — Союз: и, что, как
- PART — Частица: бы, ли, только
- P — Слово-предложение — используется для интерпретации только двух слов да и нет, способных выполнять функцию целого предложения.
- INTJ — Междометие: ого, увы, эх
- NID — Слово, представляющее собой иноязычное вкрапление в русский текст или несловесную формулу: Берлинер Цайтунг, Berliner Zeitung, Щ243.

В принципе итоговая разметка может быть сколь угодно расширена, например, в рамках данной работы для существительных дополнительно определялось его число. В качестве тренировки классификатора был выбран довольно маленький набор текстов, всего около двух мегабайт размеченных предложений. Поиск гиперплоскости занимал около 10 минут на каждый класс.

Для тестирования алгоритма было выбрано случайное подмножество предложений из корпуса. Полученные результаты сравнивались с результатом работы свободной библиотеки для морфологического анализа - pyMorph, в основе которой лежат алгоритмы исследователей группы Aot.ru. Тестирование проходило в несколько этапов: изначально тестовый набор не содержал ошибок, затем постепенно в него случайным образом вносились ошибки. Такой способ проверки работы на грязных текстах, конечно, не идеален, но ручная разметка какого-либо подходящего набора предложений занимает огромное число времени. Ниже приведена таблица полученных результатов, в колонке W приведена точность определения части речи, в колонка S содержит точность разметки предложения в целом. Под точностью подразумевается процент проходов алгоритма, в результате работы которых был выдан правильный результат.

Процент ошибок	SVM		pyMorphy	
	W	S	W	S
0%	95.7	59.0	84.2	20.6
10%	91.8	36.4	80.6	15.6
20%	88.0	26.0	77.3	11.7

Табл. 1. Результаты работы алгоритма.

Как можно заметить построенная в данной работе алгоритмическая модель работает лучше, однако динамика снижения точности при появлении ошибок примерно одинаковая. В целом алгоритм оправдывает характеристику точности.

4

Заключение

Полученный алгоритм улучшил качество определения части речи в целом по сравнению со словарными подходами, как и для текстов без ошибок, так и для текстов с ошибками. Точность работы алгоритма составляет более 90% точности при анализе текстов содержащих не более 10% ошибок. Скорость работы алгоритма, приемлема, однако существенно уступает простому подглядыванию в словарь, которое может быть осуществлено за линейное от длинные слова время. В целом увеличение скорости работы классификации посредством метода опорных векторов с учетом специфики данной задачи, можно рассматривать как отдельное направление дальнейшей работы.

Полученный алгоритм довольно гибок и может быть легко модифицирован, с целью решения каких-то специфических задач. В частности можно существует проблема нахождения нормальной формы слова, для которой информация о части речи была бы полезна. Рассмотренная в данной работе проблема может служить базисом для построения более сложных приложений обработки естественных текстов, таких как синтаксический, а затем и семантический анализ русского языка.

5

Список используемой литературы

1. Dustin Boswell, “Introduction to Support Vector Machines”, 2002
2. Jesus Gimenez and Lluis Marquez, “Fast and Accurate Part-of-Speech Tagging: The SVM Approach Revisited”, 2003.
3. Vapnik V.N., The Nature of Statistical Learning Theory. Springer, 1995
4. А.В. Сокирко, С.Ю.Толдова, “Сравнение эффективности двух методик снятия лексической и морфологической неоднозначности для русского языка (скрытая модель Маркова и синтаксический анализатор именных групп)”, - http://www.aot.ru/docs/RusCorporaHMM.htm
5. Автоматическая Обработка Текста, - http://aot.ru
6. Зализняк А.А.“Грамматический словарь русского языка” М.: Русский язык, 1980.
7. Зеленков Ю.Г., Сегалович И.В., Титов В.А., Вероятностная модель снятия морфологической омонимии на основе нормализующих подстановок и позиций соседних слов. // Компьютерная лингвистика и интеллектуальные технологии. Труды международного семинара Диалог 2005., 2005.
8. Национальный корпус русского языка, - www.ruscorpora.ru
9. Т.Ю. Кобзарева, “Морфанализ in vivo”