НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
Школа лингвистики
Факультет гуманитарных наук

КУЗНЕЦОВ ИЛЬЯ ОЛЕГОВИЧ

Автоматическая разметка семантических ролей в русском языке

Специальность 10.02.21

Прикладная и математическая лингвистика

Диссертация на соискание ученой степени

Кандидата филологических наук

Научный руководитель

к.ф.н. Бонч-Осмоловская А.А.

Москва 2016

Содержание

Введение

Автоматический анализ языка – одна из наиболее важных и перспективных задач современной вычислительной науки. Решение этой задачи – создание системы, способной понимать и порождать тексты на естественном языке – это большой шаг на пути к созданию полноценного искусственного интеллекта. Этому есть несколько причин. Во-первых, языковая коммуникация – один из основных и наиболее естественных для человека способов передачи информации, и создание прозрачного языкового интерфейса между людьми и компьютерами позволило бы значительно повысить эффективность их взаимодействия. Во-вторых, тексты традиционно используются для накопления и передачи знаний, и автоматический анализ текстов позволил бы достичь беспрецендентной эффективности в оперировании этими знаниями. В контексте развития сети Интернет и быстрого роста объёмов информации, представленной в машинночитаемом виде, эта задача приобретает ещё большую важность.

На сегодняшний день не существует систем, способных к полноценному анализу, интерпретации и порождению текстов на естественном языке, однако значительный прогресс в этой области уже достигнут. В современной вычислительной науке можно выделить две области, наиболее активно вовлечённых в исследование языка: автоматическая обработка языка (natural language processing) и компьютерная лингвистика (computational linguistics).

Компьютерная лингвистика возникла в 50е годы XX века как ответвление прикладной лингвистики. С ростом мощности и доступности компьютеров лингвистика получила дополнительный инструментарий, с помощью которого стало возможным специфицировать и верифицировать лингвистические модели и теории на больших объёмах текстовых данных. Основные задачи компьютерной лингвистики – создание лингвистических ресурсов (корпусов, словарей, тезаурусов) а также количественные и качественные исследования лингвистического материала, призванные проверить положения той или иной теории или же расширить понимание определённого лингвистического феномена.

Автоматическая обработка языка возникла как часть науки об искусственном интеллекте примерно в то же время. Непосредственной задачей автоматической обработки языка было обеспечить понимание и интерпретацию текстов на естественном языке. Первые системы автоматической обработки языка были основаны на правилах и зачастую представляли собой формализацию той или иной лингвистической или логической теории. Со временем стало понятно, что подобные системы обладают рядом недостатков, в числе которых высокая стоимоть разработки, трудность адаптации к новым языкам и новым типам текстов, недостаточная гибкость. На сегодняшний день большинство систем автоматической обработки языка основываются на статистике и машинном обучении. Основные задачи автоматической обработки языка – создание и оценка модулей анализа языка (лемматизация, морфологический анализ, синтаксический и дискурсивный анализ). В качестве обучающих и тестовых данных для систем выступают ресурсы, созданные компьютерными лингвистами, а оценка, как правило, производится на основе количественных характеристик.

Многие задачи языкового анализа на сегодняшний день переформулированы как общие задачи машинного обучения и выполняются

практически без использования знаний об исходных лингвистических моделях, на основе которых они были сформулированы. Так, например, разметка по частям речи была успешно приведена к общей задаче разметки последовательностей, а автоматический перевод зачастую выполняется с помощью общих алгоритмов выравнивания последовательностей, используемых также, например, в вычислительной биологии. Такой подход помещает автоматическую обработку языка в контекст более общей задачи обработки сигналов (signal processing), что не обязательно является адекватным подходом к анализу феномена с богатой внутренней структурой, каким является естественный язык.

Вышеуказанное наблюдение особенно актуально для задач, которые в большой степени опираются на теоретико-лингвистические модели и ограничения и основываются на структурно сложных моделях. Эти задачи находятся на стыке компьютерной лингвистики и автоматической обработки языка: успешность технического решения в большой степени зависит от того, в какой мере конечная вычислительная модель использует свойства конкретного лингвистического ресурса, лежащего в её основе, и теоретические предположения, заложенные в формализм, в соответствии с которым этот ресурс был создан. В качестве примеров таких задач можно привести синтаксический анализ, где для уменьшения сложности задачи активно используются ограничения на проективность и на структуру синтаксического дерева, и автоматическую разметку семантических ролей, в ходе которой для текста создаётся поверхностное семантическое представление на основе заранее заданного формализма и которой посвящена настоящая работа.

Автоматическая разметка семантических ролей, или **автоматическая разметка актантов** (*semantic role labeling*, *SRL*) – одно из приоритетных направлений в современной автоматической обработке языка. Это тип высокоуровневого анализа текста, при котором для исходного текста на

естественном языке порождается поверхностная интерпретация на основе теории семантических ролей.

Рассмотрим следующий пример. Предположим, что нам дано предложение на естественном языке,и в этом предложении выбран некоторый **предикат** (например, глагол). Задача автоматической разметки актантов состоит в том, чтобы найти **актантов**, т.е. участников ситуации описанной данным предикатом, и приписать им **семантические роли**. Так, например, предложение "Пётр купил яблоко за 5 рублей" будет проанализировано следующим образом:

[Пётр]$_{\text{Покупатель}}$ купил [яблоко]$_{\text{Товар}}$ за [5 рублей]$_{\text{Цена}}$

Пример 1: Разбор предложения в терминах семантических ролей

Отметим, что автоматическая разметка актантов отличается от синтаксического парсинга, в ходе которого анализу подвергается грамматика, а не семантика высказывания. В ходе процедуры синтаксического разбора предложения слова объединяются в синтаксические группы (в случае анализа в терминах непосредственных составляющих) или между ними устанавливаются синтаксические связи (в случае, если парсер опирается на формализм деревьев зависимостей). Несмотря на наличие определенных корреляций между семантическими ролями и синтаксическим оформлением участников ситуации, эти явления не эквивалентны и относятся к разным уровням языковой модели. Синтаксический анализ – строгая процедура, которая опирается на грамматику языка и в большинстве случаев подразумевает единственный правильный результат анализа. Автоматическая разметка актантов — гораздо более субъективная задача, в которой большую роль играет интерпретация ситуации человеком.

В то же время следует понимать, что автоматическая разметка актантов – это не полный семантический анализ, т.к. работа всегда производится на уровне предложения, и системы не используют правил логического вывода. Результат автоматической разметки актантов — не полное семантическое представление исходного предложения, а в большей степени поверхностный рефлекс этого семантического представления, который, несмотря на свою неполноту, оказывается полезен при решении ряда прикладных задач. Важность выбранной нами темы связана в первую очередь с тем, что анализ текста в терминах семантических ролей позволяет сравнительно небольшими усилиями получить дополнительный уровень абстракции, описывающий семантику текста. Информация о семантических ролях может быть затем использована для извлечения фактов [Christensen, Soderland, Etzioni, 2010], машинного перевода [Liu, Gildea, 2010], в вопросно-ответных системах [Shen, Lapata, 2007], а также, потенциально, в любой системе автоматической обработки языка, которая так или иначе опирается на семантическую информацию.

Автоматическая разметка актантов в современном понимании возникла в начале 2000х годов [Gildea, Jurafsky, 2000]. Теоретической основой для направления послужила **теория семантических ролей** Ч. Филлмора [Fillmore, 1968]. Прикладным основанием экспериментов в этой области можно считать построенные на базе теории Филлмора лексико-грамматические ресурсы: и её ответвления (в первую очередь, модели FrameNet [Baker, Fillmore, Lowe, 1998], PropBank [Palmer, Gildea, Kingsbury, 2005] и VerbNet [Schuler, 2005]). Теория семантических ролей описывает ролевые инвентари и задаёт общую семантическую модель, на основе которой производится анализ ситуаций.

Первые системы автоматической разметки актантов были созданы для английского языка, который на тот момент обладал наиболее обширными ресурсами и развитой инфраструктурой. Со временем ресурсы стали

создаваться и для других языков, однако английский язык до сих пор сохраняет первенство в плане качества разрабатываемых систем и их применения в реальных приложениях. Исторически многие методы автоматической обработки языка были созданы на базе английского и затем перенесены на другие языки. В то же время по очевидным причинам прямой перенос методов и систем между языками невозможен: каждый язык обладает уникальными особенностями, и зачастую даже используемые алгоритмы требуют значительной модификации, прежде чем аналогичный английскому инструмент сможет быть использован для других языков. Среди ярких примеров таких отличий – автоматический анализ морфологии, который для английского языка сводится к определению частей речи и успешно выполняется с помощью простейших моделей, в то время как для языков с богатой морфологией требуется анализ и снятие неоднозначности на символьном уровне. Другой пример – синтаксический анализ, который в английском языке в первую очередь опирается на порядок слов и части речи, однако в языках со свободным порядком слов и развитым элилпсисом для решения этой задачи требуются значительно более сложные и гибкие модели. Было неоднократно продемонстрировано, что системы автоматической разметки актантов также теряют в качестве при переносе на другой язык [Björkelund, Hafdell, Nugues, 2009]. В дальнейшем для отсылки к этой проблеме мы будем использовать понятие **языковой специфичности**.

Другая причина, по которой системы автоматического выделения глагольных актантов для языков, отличных от английского, отстают от английских систем – доступность ресурсов. Исторически первые системы автоматической разметки актантов были основаны на правилах [Hirst, 1988]. Эти системы сильно отличались от современных, т.к. были ориентированы на анализ текстов из узких предметных областей и оперировали специфичными

наборами семантических ролей, которые зачастую были мотивированы прикладными задачами, а не лингвистической теорией.

Большинство современных систем SRL основаны на машинном обучении с учителем: система автоматически обучается выполнять задачу на основе размеченного **корпуса примеров**. Создание такого корпуса – крайне трудоёмкая задача, и подобные ресурсы существуют лишь для ограниченного числа языков. Для обозначения комплекса проблем, связанных с недостатком ресурсов, мы будем использовать понятие **ресурсозависимости**.

В последние годы было проведено множество исследований по автоматической обработке текстов для русского языка. Так, в 2010 году прошло соревнование морфологических анализаторов [Ляшевская и др., 2010] в 2012 – соревнование синтаксических парсеров [Толдова и др., 2012], в 2014 – соревнование систем разрешения анафоры [Toldova и др., 2014].

Несмотря на общую популярность, тема автоматической разметки актантов почти не исследовалась на русском материале, и одной из причин этого было отсутствие обучающего и тестового корпуса. Единственным подходящим ресурсом для русского языка на сегодняшний день является FrameBank, один из компонентов которого представляет собой корпус с необходимой для нашей задачи разметкой. Помимо корпуса, ресурс включает в себя описание конструкций с различными глаголами и другую лексикографическую информацию (подробнее см. [Ляшевская, Кузнецова, 2009]). В рамках диссертационного исследования мы разработали систему автоматической разметки актантов, опираясь на промежуточную версию этого ресурса. Подобной работы на материале FrameBank ранее не проводилось.

Объект нашего исследования – автоматическая разметка актантов для методами машинного обучения для русского языка. Цель исследования – разработать и описать систему автоматической разметки актантов и детально изучить результаты её работы, выяснить вклад различных лингвистических

свойств и других параметров задачи в качество классификации. В качестве материала исследование опирается на корпус примеров FrameBank, а также на построенные на основе этого корпуса модели. Автоматическая разметка актантов для русского языка – одно из наименее развитых направлений в автоматической обработке текста, что, учитывая большое прикладное значение этой задачи, объясняет её актуальность. Научная новизна работы состоит в том, что ранее подобных исследований на русском материале не проводилось. Предложенное исследование – первый опыт применения систем на основе машинного обучения к корпусу примеров FrameBank. Ряд частных решений также применяется к русскому языку впервые, кроме того, это первое известное нам полноценное описание подобной системы, достаточно подробное для успешной реимплементации и усовершенствования предложенного метода. Теоретическая значимость исследования состоит в оценке вклада различных лингвистических свойств в качество работы классификатора. Мы предлагаем и подробно анализируем ряд свойств, которые по причинам типологического характера не могут быть использованы на английском материале и потому почти не представлены в литературе. Практическая значимость исследования состоит в подробном качественном и количественном анализе результатов работы системы. Кроме того, работа содержит детальное описание компонентов системы, а также ряд рекомендаций по усовершенствованию ресурса, основанных на нашем опыте, которые помогут усвершенствовать ресурс и сделать исследования на его основе более доступными.

Диссертация состоит из введения, четырёх глав, заключения и библиографии. В **Главе I** задача автоматической разметки семантических ролей рассматривается в исторической перспективе. Как упоминалось выше, автоматическая разметка актантов – одна из наболее теоретически вовлеченных задач в автоматической обработке языка, и кажется разумным

подробно остановиться на теоретической стороне задачи, чтобы мотивировать решения и ограничения, которые мы принимаем на этапе практической реализации системы. Также глава содержит обзор и историю развития подходов к автоматическому выделению семантических ролей, начиная от первых работ, опубликованных в начале 2000-х годов, и заканчивая наиболее современными системами на основе частичного обучения с учителем и обучения без учителя. **Глава II** посвящена описанию разработанной системы. **Глава III** рассказывает в метриках и процедуре оценки качества, а также содержит анализ результатов работы системы. **Глава IV** подводит итоги работы и определяет дальнейшие пути развития автоматической разметки актантов и сопутствующих ресурсов применительно к русскому языку на основании приобретённого нами опыта.

I. Теория семантических ролей и автоматическая разметка актантов

I.1 Теоретические основы

В теоретическом отношении автоматическая обработка актантов опирается на теорию семантических ролей. Исторически понятие семантической роли в том или ином виде присутствовало в большинстве лингвистических теорий, однако несмотря на то, что этот концепт, как правило, интуитивно понятен, и существование семантических ролей не подвергается сомнению, до сих пор ведутся споры о том, как именно следует определять семантическую роль, каков инвентарь этих ролей, каково место семантических ролей в системе языка и какие функции они выполняют. Поэтому прежде чем перейти к непосредственно решению задачи автоматической обработки актантов, кажется уместным ненадолго остановиться на теории семантических ролей, истории её развития и современных направлениях исследований в данной области.

Традиционно первым упоминанием семантических ролей принято считать систему падежей kāraka, предложенную Панини для описания грамматики санскрита [Misra, 1966]. Kāraka определяется как семантическое отношение между глаголом и зависимым именем, которое обусловливает морфологическую форму имени. Панини использует 6 падежей-kāraka – агент, объект, инструмент, пункт назначения, источник и локус – которым в санскрите соответствуют падежи – номинатив, аккузатив, инструменталис, датив, аблатив и локатив соответственно. Соотношение между kāraka и морфологическими падежами не было однозначным, так, например, в конструкции с пассивным глаголом агент маркируется инструменталисом, однако сохраняет свою агентивную kāraka-роль. Несмотря на очевидное сходство с понятием глубинного падежа, падежи-kāraka в системе Панини таковыми не являлись и использовались скорее как средство описания объективной реальности. Работы Панини не имели большого влияния на западную лингвистическую традицию, хотя и были в целом хорошо известны [Malchukov, Spencer, 2012].

Понятие семантической роли, которое используется в современной автоматической обработке актантов, основывается на работах Ч. Филлмора [Fillmore, 1968], который, собственно, и ввёл понятие семантической роли в современный лингвистический дискурс, и Дж. Грубера [Gruber, 1965], который оперировал концептуально схожим понятием тематического отношения. Изначально основной задачей аппарата семантических ролей было описание ограничений, налагаемых глаголом (или, более широко, предикатом) на количество и состав аргументов. В ранних версиях порождающей грамматики [Chomsky, 1965] подобные ограничения задавались рамкой субкатегаризации глагола (см. Прим. 2), которая задавалась отдельно для каждого глагола в качестве лексической информации.

Kill: [NP kill NP]

Drink: [NP drink (NP)]

Search: [NP look (for NP)]

Пример 2: Рамки субкатегоризации

Подобный подход, однако, был неэкономичным с описательной точки зрения. Кроме того, было отмечено, что существует ограниченное количество рамок субкатегоризации, что семантически сходные глаголы имеют сходные рамки субкатегоризации, и, наконец, что ограничения на рамки и набор доступных для глаголов трансформаций допускают определённые обобщения. В более поздних версиях порождающей грамматики [Chomsky, 1982] для описания подобных ограничений используется аппарат тета-ролей: каждому предикату приписывается т.н. тета-сетка (theta-grid, см. Прим. 3), которая задаёт ограничения на набор и состав аргументов в терминах тета-ролей (которые в целом соответствуют семантическим ролям). Грамматичность предложения затем проверяется на основании тета-критерия: каждой тета-роли, указанной в тета-сетке, должен соответствовать один аргумент, и каждому аргументу может быть приписана только одна тета-роль. Данный подход позволяет делать обобщения о синтаксическом поведении различных глаголов.

give<source(DP), theme(DP), goal(PP)>

Пример 3: Тета-сетка

В то же время следует отметить, что тета-роли представляют собой синтаксический конструкт, в то время как семантические роли в Филлморовском понимании описывают в первую очередь семантику предиката.

Классическая теория семантических ролей, изложенная в [Fillmore, 1968], постулирует наличие набора семантических ролей, обладающих следующими свойствами:

- **Полнота** и **уникальность** – *каждый* аргумент глагола имеет *ровно одну* семантическую роль
- **Единственность заполнения** роли – каждая роль может быть заполнена только один раз
- **Независимость** и **атомарность** – определение семантической роли не должно зависеть от конкретного выбранного предиката и от других ролей. Семантическая роль имеет категориальную природу и не может быть разделена на компоненты.

На основании этих критериев Ч. Филлмор предложил следующий классический инвентарь ролей:

- **Агенс** – одушевленный инициатор события, способный по своей воле его прекратить
- **Пациенс** – партиципант, наиболее вовлеченный в событие и претерпевающий наиболее значительные изменения
- **Бенефактив** – участник, чьи интересы наиболее затронуты в ходе ситуации
- **Экспериенцер** – получатель информации при глаголах восприятия
- **Стимул** – источник информации при глаголах восприятия
- **Инструмент** – неодушевленный объект, с помощью которого осуществляется действие, но который при этом не претерпевает изменений
- **Адресат** – получатель сообщения при глаголах речи
- **Источник** – место, из которого осуществляется движение

- **Цель** – место, в которое осуществляется движение

Семантическая роль в классическом подходе характеризуется, с одной стороны, синтаксическим оформлением, а с другой – лексическими ограничениями на заполнение роли. Одно из наиболее важных свойств семантических ролей в прикладном отношении – устойчивость к трансформациям, например:

[Иван]$_{\text{Агенс}}$ сломал [стол]$_{\text{Пациенс}}$

[Стол]$_{\text{Пациенс}}$ был сломан [Иваном]$_{\text{Агенс}}$

Пример 4: Устойчивость семантических ролей к трансформациям

В ходе дальнейших исследований семантических ролей выяснилось, однако, что предложенный Ч. Филлмором инвентарь обладает ограниченными описательными возможностями, и что ни одно из указанных выше свойств не является абсолютным. Основные проблемы, с которыми столкнулась теория – это проблема **фрагментации ролей** [Dowty, 1991] и проблема **неструктурированности ролевого инвентаря** [Jackendoff, 1983]. Проблема фрагментации ролей связана с тем, что для повышения описательной силы теории при соблюдении теоретических требований к ролям приходится вводить новые семантические роли, что, в свою очередь, приводит к снижению описательной силы теории. Рассмотрим следующие предложения (Прим. 5):

[Иван]$_{\text{Агенс}}$ готовит [мясо]$_{\text{Пациенс}}$ на [огне]$_{\text{Инструмент}}$

[Иван]$_{\text{Агенс}}$ готовит [мясо]$_{\text{Пациенс}}$ [в котле]$_{\text{Инструмент}}$

*[Иван]$_{\text{Агенс}}$ готовит [мясо]$_{\text{Пациенс}}$ [в котле]$_{\text{Инструмент}}$ [на сковородке]$_{\text{!Инструмент!}}$

[Иван]$_{\text{Агенс}}$ готовит [мясо]$_{\text{Пациенс}}$ в [котле]$_{\text{Инструмент}}$ на [огне]$_{??}$

Пример 5: Трудности с разграничением ролей

Для того чтобы принцип единственности заполнения роли (2) соблюдался, нам в данном случае требуется ввести новую роль. Однако в этом случае данная роль поступает непосредственно в наш ролевой инвентарь и в соответствии с (3) получает универсальное определение, в результате чего ролевой инвентарь растёт.

Другая проблема классической теории семантических ролей – отсутствие внутренней организации в ролевом инвентаре. Так, как, например, отмечает Р. Джекендофф на примере глагола *come*, многие глаголы движения принимают в качестве семантического аргумента "Путь" в его различных конфигурациях, однако описательных возможностей стандартного инвентаря семантических ролей недостаточно для регистрации этого явления, см. пример из [Jackendoff, 1983]:

Pat came [to the library]$_{\text{Цель}}$

Pat came [from the cafeteria]$_{\text{Источник}}$

Pat came [to the library]$_{\text{Источник}}$ [from the cafeteria]$_{\text{Цель}}$

Пример 6: Варианты ролевого маркирования пути

В данном случае для описания ролевого набора глаголов движения потребовалось бы либо ввести новую независимую роль Путь (нарушив, тем самым, критерий (3)), либо включить дополнительный уровень иерархии, который свидетельствовал бы о том, что роли "Цель" и "Источник" могут выступать компонентом "Пути".

На сегодняшний день существует три основных подхода к созданию инвентаря категориальных (т.е. неделимых) семантических ролей. Первый подход использует наиболее дробное представление ролей, в котором роли являются **предикатно-специфическими**, т.е. уникальными для каждого предиката: например, у глагола "*убивать*" будут представлены роли "*тот,*

кто убивает", "*тот, кого убивают*", "*орудие убийства*" и т.д. При таком подходе описание общих свойств актантов различных глаголов становится затруднительным, однако описание лексических ограничений не представляет трудностей. На другом конце спектра находятся подходы, опирающиеся на максимально **обобщённые роли** Актора и Претерпевающего: эти роли отвечают за большую долю вариативности в синтаксическом поведении аргументов, и использование крупных ролей открывает возможности для генерализации, недоступные для более "дробных" инвентарей, в то же время понижая внутреннюю семантическую однородность ролей. Наконец, в середине спектра находятся **классические ролевые инвентари** наподобие предложенного Ч. Филлмором. Схема, составленная Р. Ван Валином [Van Valin, 1999] демонстрирует соответствие ролей в данных типах инвентарей.

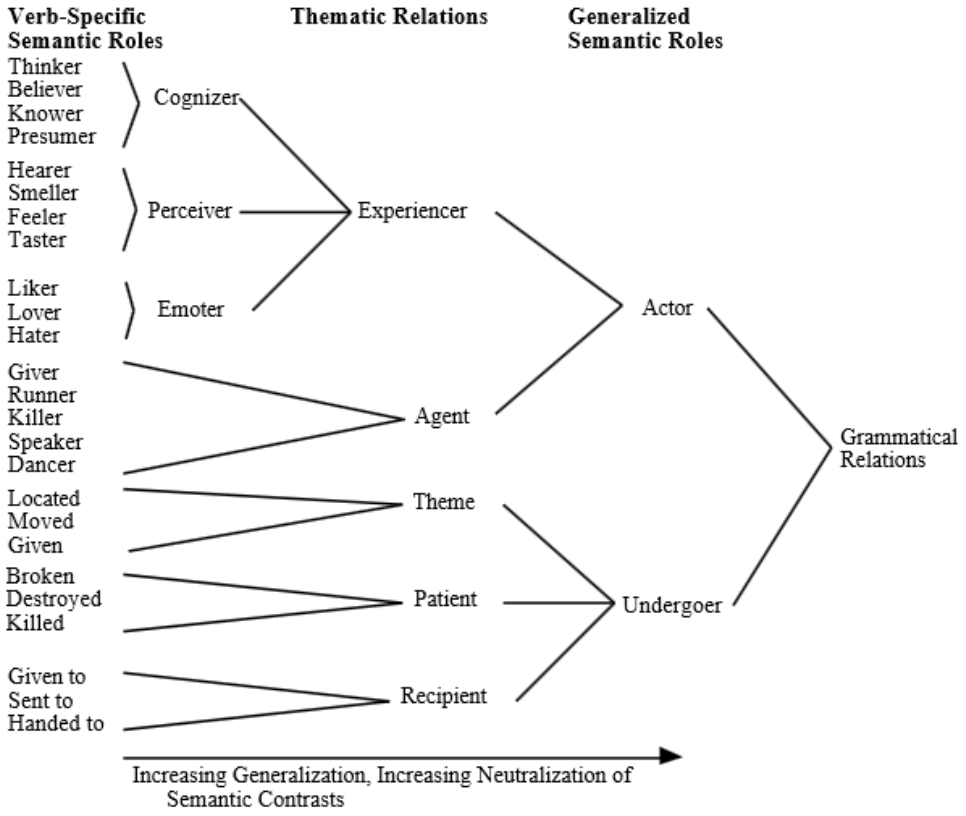

Рисунок 1: Иерархия типов семантических ролей (Р. Ван Валин)

В контексте автоматической разметки актантов наибольшую популярность имеют подходы на основе предикатно-специфических ролей. Связано это, в первую очередь, с тем, что два основных корпуса, использующихся для обучения систем semantic role labeling, используют дробные роли для описания глагольной семантики. Наибольшей популярностью пользуется корпус **PropBank** [Palmer, Gildea, Kingsbury, 2005], в котором роли определяются независимо для каждого предиката. Это обусловлено, как нам кажется, причинами прикладного характера: корпус PropBank обладает наибольшим покрытием среди доступных корпусов, аннотированных семантическими ролями, а также содержит синтаксическую разметку, что значительно сокращает затраты на разработку системы. Другой популярный ресурс – корпус примеров, сопровождающий лексико-семантический ресурс **FrameNet** [Baker, Fillmore, Lowe, 1998]. FrameNet описывает значение предикатов в рамках фреймовой семантики: предикаты группируются по ситуациям-фреймам, и для каждого фрейма используется свой ролевой набор, при этом как фреймы, так и роли допускают наследование, т.е. ролевой инвентарь FrameNet структурирован. Подобный подход достаточно гибок в теоретическом отношении, т.к. позволяет использовать нужный уровень специфичности в зависимости от поставленных задач, однако на практике FrameNet крайне неоднороден с точки зрения специфичности/абстрактности используемых ролей, что затрудняет разработку систем автоматической разметки актантов на данном материале и использование их для решения практических задач.

Несмотря на продолжающиеся споры о том, какой из подходов позволяет наиболее адекватно описать семантические роли, кажется достойным внимания и то, как исследователи из различных областей лингвистики и различных лингвистических традиций приходят к крайне похожим выводам относительно теории семантических ролей и насколько похожими оказываются

конструкты, полученные в результате их теоретических исследований. Параллельно с возникновением и развитием филлморовской теории семантических ролей, призванной решить некоторые из проблем, возникших в рамках генеративной грамматики, в Московской семантической школе (МСШ) был предложен аппарат моделей управления, выполнявший во многом схожие функции, но направленный на решение проблем, связанных с заполнением синтаксических и семантических валентностей предиката [Апресян, 1974].

В рамках теории МСШ каждой лексеме соответствует **толкование**. В толкование некоторых лексем входят переменные, которые также называются **семантическими валентностями** лексемы (термин "валентность" восходит к работам Л. Теньера [Теньер, 1988]) , сама лексема в этом случае называется предикатом. Предикат не обязательно должен быть глаголом: так, например, семантическими валентностями обладают отглагольные существительные и имена родства (*убийство* – кем, кого; *сын* – чей). Кроме того, каждая лексема имеет набор **синтаксических валентностей**, определяющих, какие слова или категории данная лексема может иметь в качестве вершины и зависимых. Часть синтаксических валентностей обусловлена частеречной принадлежностью слова (например, глагол всегда может иметь зависимое-наречие), другие же, однако, являются уникальными для выбранной лексемы и соответствуют семантическим валентностям этой лексемы.

Соответствие семантических и субкатегориальных, т.е. не обусловленных категорией, синтаксических валентностей закрепляется в **модели управления** (МУ) предиката, которая вместе с непосредственно толкованием характеризует лексему. Например, толкование и модель управления для глагола "*курить*" будет выглядеть следующим образом [Мельчук, Жолковский, 1984]:

X курит Y = X вдыхает через рот непосредственно из специального устройства Y дым тлеющего в Y-е вещества Z, обычно – табака, и вдыхает этот дым – с целью, чтобы это каузировало то, что X испытывает приятное ощущение

1=X [кто вдыхает]	2=Y [из какого устройства]	3=Z [какое вещество тлеет
сущ. им.п.	сущ. в.п.	--
сущ. им.п.	сущ. в.п.	из сущ. р.п

Пример 7: Толкование и модель управления

Обратим внимание, что заполнение всех семантических валентностей требуется не всегда: так, например, одна из моделей управления глагола "*курить*", представленных выше, допускает опущение третьего участника.

Модель управления – компактный и удобный способ кодирования соответствия между семантическими и синтаксическими валентностями предиката. В рамках этого формализма не все синтаксические валентности должны быть связаны с семантическими валентностями: в случае, когда это так, единица, заполняющая синтаксическую и семантическую валентность, называется **актантом** выбранного предиката. В случаях, когда это не так, т.е. выбранной единице не соответствует никакая из семантических валентностей, она объявляется **сирконстантом**. Так, например, в предложении "*Петя курит трубку на улице*" "*Петя*" и "*Трубка*" являются актантами, т.к. входят в толкование лексемы "*курить*", "*на улице*" же, хотя и синтаксически зависит от предиката, является сирконстантом.

Несмотря на то, что формализм моделей управления в теоретическом отношении отличается от формализма семантических ролей, мы можем заметить их функциональное и описательное сходство. Актанты кодируются в толкованиях уникальными предикатно-специфическими буквенными обозначениями, которые сохраняют своё значение при трансформациях, что

делает их функционально эквивалентными семантическим ролям в предикатно-специфическом смысле. Как и в случае с семантическими ролями, соблюдается ограничение на единственность заполнения актанта, а сами переменные-валентности атомарны и неделимы.

Именно формализм МСШ использовался при разметке корпуса примеров **FrameBank** [Lyashevskaya, Kashkin, 2015] – единственного доступного на сегодняшний день корпусного ресурса с разметкой, подходящей для обучения систем автоматической классификации актантов в русском языке. Исходя из того, что понятие актанта в МСШ и понятие предикатно-специфичной семантической роли в ресурсах типа PropBank функционально эквивалентны, мы ставим перед собой задачу автоматической разметки актантов – или автоматической разметки семантических ролей, и в дальнейшем используем два этих понятия как взаимозаменяемые, хотя с теоретической точки зрения это не совсем соответствует действительности. При разработке системы автоматческой разметки актантов мы опираемся на характеристики семантических ролей, которые традиционно используются в semantic role labeling, и моделируем синтаксическое оформление актантов, ограничения на лексическое заполнение валентностей, устойчивость к трансформациям и ограничение на единственность заполнения роли.

Следует отметить, что для FrameBank также разрабатывается иерархия семантических ролей, что позволит в дальнейшем использовать этот ресурс для работы с более абстрактными ролями. В данной работе, однако, мы сосредоточили внимание на предикатно-специфических ролях.

Прежде чем перейти к описанию прикладного аспекта исследования, кажется важным остановить внимание на ещё одном теоретически перспективном направлении в выбранной нами области.

Как уже упоминалось выше, абсолютное большинство современных систем semantic role labeling опираются на корпуса, созданные на основе

категориальных теорий семантических ролей. В категориальных теориях постулируются жёсткие границы между семантическими ролями и роли объявляются неделимыми, что влечёт за собой массу сложностей, связанных с определением инвентаря и растущей гранулярностью ролей. В то же время категориальность не является обязательным условием, и один из наиболее перспективных на сегодняшний день теоретических подходов к описанию семантических ролей отказывается от этого свойства. Так, Д. Даути [Dowty, 1991] предлагает вместо опоры на жесткие категории использовать для описания семантических ролей ряд признаков, принадлежащих к **прото-ролям** Прото-Агенса и Прото-Пациенса. **Прото-Агенс** (1) волитивно вовлечён в событие, (2) сознателен/воспринимает событие, (3) инициирует событие, (4) движется и (5) существует независимо от события. **Прото-Пациенс**, в свою очередь, (1) претерпевает изменение, (2) подвергается каузации, (3) неподвижен по отношению к другому участнику события, (4) является инкрементальной темой и (5) не существует независимо от события. Даути в своей работе подчёркивает, что прото-роли не реализуются в действительности и являются лишь прототипами, на основании свойств которых различаются роли конкретных предикатов. Подобный подход позволяет решить проблему фрагментированности пространства семантических ролей, при этом не уменьшая способности теории к генерализации.

На сегодняшний день существует только одна инициатива по разметке корпуса с использованием ролевых свойств вместо категориальных ролей [Reisinger, Rawlins, Durme, 2015] и обучению систем автоматчиеской разметки актантов на основе этих данных. Данная тема представляет большой интерес, и мы рассчитываем увидеть работы, посвящённые этому вопросу, в ближайшем будущем. Следует отметить, что автоматическое распознавание признаков может оказаться более сложной задачей, чем автоматическая разметка актантов, и что интеграция результатов работы подобной системы в конечные

приложения – отдельная прикладная задача. Данный пример хорошо иллюстрирует неизбежное отставание прикладных методов, основанных на размеченных вручную массивах данных, от прогресса в теоретической лингвистике, и демонстрирует важность теоретических исследований для автоматической разметки актантов и для прикладной лингвистики в целом.

I.2 Задача и мотивация

Последнее десятилетие характеризуется ростом внимания к концепции семантических ролей в контексте автоматической обработки естественного языка. Основное направление исследований в этой области — автоматическая классификация актантов (semantic role labeling). Задача автоматической классификации актантов определяется следующим образом. Предположим, что нам надо предложение на естественном языке, и в этом предложении выбран целевой предикат (например, глагол). Требуется найти в предложении актанты этого предиката и присвоить этим актантам семантические роли [Gildea, Jurafsky, 2000].

Тема автоматической классификации актантов достаточно популярна в современной компьютерной лингвистике: ежегодно появляется множество работ, посвящённых этому вопросу, регулярно проводятся крупные соревнования систем (например, Senseval-3 в 2004-м году [Litkowski, 2004], CoNLL в 2005, 2008 и 2009-м [Carreras, Marquez, 2005; Hajič и др., 2009; Surdeanu и др., 2008]). Для английского языка уже достигнуты приемлемые результаты, которые позволяют использовать технологию в промышленных системах [Björkelund, Hafdell, Nugues, 2009; Das и др., 2010]

Одна из причин популярности semantic role labeling, по всей видимости, состоит в том, что аппарат семантических ролей позволяет создать уровень семантической интерпретации, достаточный для решения многих прикладных задач. Так, автоматическая разметка актантов зачастую позволяет ответить на классический набор вопросов "*кто*", "*что*", "*где*", "*когда*", "*почему*", что оказывается полезным при разработке вопросно-ответных систем [Shen, Lapata, 2007], систем извлечения информации, в частности извлечения фактов [Christensen, Soderland, Etzioni, 2010] и других приложениий. Представление на

основе семантических ролей успешно применяется для снятия омонимии, при решении задач машинного перевода [Liu, Gildea, 2010] и может быть использовано почти в любой задаче по автоматической обработке языка, где требуется семантический анализ.

Можно рассматривать автоматическую разметку актантов как компромисс между качеством анализа и ценностью результатов: синтаксический анализ не предоставляет семантической информации в явном виде, а глубинный семантический анализ с использованием логики и баз знаний слишком сложен и на сегодняшний день полностью не реализован. Автоматическая разметка актантов позволяет получить поверхностный семантический анализ предложения с приемлемым качеством.

Автоматическая разметка актантов как задача приобрела свой современный вид в начале 2000-х годов. Ранние системы, выполнявшие аналогичные функции, основывались на правилах и шаблонах. Основной проблемой систем на основе **правилового подхода** была узкая специализация и высокие затраты на разработку, т.к. правила составлялись вручную для небольших групп предикатов в рамках узких тематических областей (существуют и современные системы, основанных на правилах, см. например, [Anisimovich и др., 2012]).

На сегодняшний день большинство систем автоматической разметки актантов основывается на **машинном обучении**, что позволяет избежать вышеупомянутых проблем правилового подхода – в первую очередь, здесь имеется в виду зависимость от языка, сложность адаптации и высокие трудозатраты на разработку. Системы машинного обучения приписывают **экземплярам**, описанным в терминах **свойств** (*features*), метку **целевого класса**. Задача машинного обучения – на основании **трерировочного набора** экземпляров, для которых значение класса известно, построить **решающую функцию**, которая будет приписывать метку класса новым экземплярам.

Поскольку для машинного обучения необходимы размеченные вручную данные, создание подобных систем стало возможным лишь после того, как появились наборы обучающих данных FrameNet [Baker, Fillmore, Lowe, 1998], PropBank [Palmer, Gildea, Kingsbury, 2005], NomBank [Meyers, 2007] и другие.

Одной из первых публикаций, посвящённых автоматической обработке актантов в её современном виде, стала статья Д. Журафски и Д. Гилдеа [Gildea, Jurafsky, 2000]. Эта работа во многом определила путь, по которому стало развиваться рассматриваемое направление. Автоматическая разметка актантов была сформулирована как задача классификации, в которой отрезкам исходного предложения требуется приписать семантические метки или роли. Работа была выполнена для английского языка на основе корпуса FrameNet и деревьев непосредственных составляющих. Авторы разделили задачу автоматической разметки актантов на два этапа: определение актантов, т. е. синтаксических групп, которые так или иначе относятся в выбранному предикату, и классификацию актантов, при которой выбранным группам приписываются роли. Подобное разделение Гилдеа и Журафски мотивировали тем, что для решения этих задач используются различные наборы свойств, которые определяют решение классификатора. В качестве свойств для обучения использовались свойства на основе синтаксиса (тип составляющей, позиция относительно предиката, путь до предиката в дереве составляющих) и семантики (лемма для терминальных узлов, значение слова на основе тезауруса). Авторы отмечают роль синтаксиса в автоматической классификации актантов, а также указывают, что использование внешних семантических ресурсов повышает полноту системы, позволяя ей работать со словами, которые не представлены в обучающем корпусе.

I.3 История автоматической разметки актантов

Работа Д. Гилдеа и Д. Журафски, посвящённая автоматическому извлечению семантических ролей, вызвала большой резонанс в научном сообществе, и задача semantic role labeling стала одной из центральных задач автоматической обработки языка на следующие годы. На сегодняшний день системы автоматической разметки актантов для английского языка демонстрируют высокое качество работы. В последние годы фокус исследований в этой области сместился в сторону систем автоматической классификации актантов на основе частичного обучения с учителем и систем обучения без учителя. Ниже мы рассмотрим основные этапы развития подходов к задаче автоматической разметке семантических ролей и подробнее остановимся на некоторых системах, которые кажутся нам ключевыми, представляют интерес с точки зрения нашей работы или же кажутся полезными в контексте развития автоматической обработки актантов для русского языка.

Параллельно с работой Д. Гилдеа и Д. Журафски, посвящённой автоматической разметке актантов с использованием ролей FrameNet, увидела свет работа Д. Гилдеа и М. Палмер [Gildea, Palmer, 2002], посвящённая разметке семантических ролей на основе корпуса PropBank с сопоставимыми результатами.

В 2004 и 2005 годах в рамках конференции CoNLL были проведены соревнования по автоматической разметке актантов [Carreras, Marquez, 2005]. В рамках соревнований автоматическая разметка актантов производилась на материале английского языка с использованием **синтаксиса непосредственных составляющих**. В качестве исходных данных использовался корпус PropBank, в котором каждому предложению сопоставлена разметка предикатно-специфическими ролями. Корпус PropBank имеет ряд особенностей, которые

сделали его более привлекательным для исследований по автоматической разметке актантов на ранних этапах. Во-первых, PropBank создан на основе синтаксического корпуса, другими словами, обучающая выборка, полученная из PropBank уже включает в себя вручную размеченные синтаксические деревья. Это избавило участников соревнований от необходимости включать в систему внешний парсер, и значительно упростило задачу в целом, поскольку качество автоматической разметки актантов в большой степени зависит от качества синтаксического анализа. Во-вторых, семантическая разметка PropBank опирается на синтаксис и в том отношении, что границы семантических аннотаций в целом совпадают с границами групп. Иначе говоря, для участников соревнований снималась задача соотнесения семантической разметки с единицами уровня синтаксиса. Таким образом, задача semantic role labeling на совернованиях CoNLL 2004 и 2005 состояла в обнаружении и классификации синтаксических групп, которые относятся к актантам того или иного предиката.

Лучший результат на соревновании CoNLL-2005 продемонстрировала система В. Пуньяканок [Koomen и др., 2005] с F1-мерой равной 79.44. Архитектура предложенной в этой работе системы состоит из трёх модулей: идентификации актантов, присвоения ролей и дополнительного модуля глобальной оптимизации. На этапе идентификации актантов производится бинарная классификация синтаксических групп на основании стандартного набора свойств. В результате этой классификации для каждого узла исходного дерева составляющих принимается решение о том, является ли он актантом выбранного предиката-цели. На этапе присвоения ролей каждый узел, выбранный в качестве актанта, получает класс – семантическую роль из заранее заданного набора (который включает в себя роли для данного предиката, а также специальный класс None, обозначающий отсутствие роли). Классификация на обоих этапах производилась с помощью алгоритма SNOW

(вариация нейронных сетей, [Roth, 1998]). Наконец, на этапе **глобальной оптимизации** решения классификаторов дополнительно обрабатываются с помощью метода целочисленного программирования. В результате этого выбирается комбинация решений, при которой актанты не пересекаются, для каждого предиката каждая роль заполняется только один раз и полученная комбинация максимизирует суммарный вес классов. Следует отметить, что рассматриваемая система одной из первых использовала дополнительный модуль глобальной оптимизации.

Другая интересная работа, также представленная в рамках CoNLL-2005 – исследование М. Сурдеану и Дж. Турмо [Surdeanu, Turmo, 2005], посвящённое сравнению качества работы систем SRL на основе полного и частичного синтаксического разбора. На тот момент существовало два основных подхода к синтаксической предобработке данных для автоматической классификации актантов. В первом случае в качестве синтаксической информации системе передавалось полное дерево непосредственных составляющих. Интуитивно такой подход кажется правильным, т.к. система получает больше информации на вход, однако на практике из-за ошибок синтаксического анализатора информация о синтаксической структуре могла быть сильно искажена, что отрицательно влияло на результат работы системы. В качестве альтернативного решения предлагалось использовать частичный синтаксический анализ, который разбивал бы клаузу на последовательность синтаксически цельных отрезков (*chunks*) [Pradhan и др., 2005]. Работа М. Сурдеану и Дж. Турмо показала, что несмотря на ошибки синтаксического анализатора, использование полного синтаксического анализа позволяет получить лучшие или по крайней мере сопоставимые результаты. В качестве классификатора использовался AdaBoost-ансамбль [Schapire, 1999] на основе одноуровневых деревьев принятия решений. Классификация производилось независимо для каждой роли, т. е. без использования модуля глобальной оптимизации.

Работа [Pradhan и др., 2005] демонстрирует альтернативный подход, в котором semantic role labeling интерпретируется как задача сегментации. Для решения задачи авторы использовали классификатор на основе метода опорных векторов (*Support Vector Machine*, *SVM*), предварительно трансформировав исходные данные с использованием BIO-нотации [Ramshaw, Marcus, 1995], в которой слова текста размечаются как начинающие семантическую роль (*Begin*), находящиеся внутри роли (Inside) и завершающие роль (*Outside*). Подобный подход, при котором автоматическая разметка актантов интерпретируется как задача сегментации, был также применён в работе [Màrquez и др., 2005], которая целиком посвящена этому вопросу и содержит более детальный анализ поведения BIO-классификаторов в зависимости от выбора метода сегментации и синтаксической структуры, на основе которой проводится сегментация.

Наконец, кажется важным упомянуть работу [Ngai и др., 2004], в которой авторы произвели сравнение пяти наиболее популярных на тот момент в рамках SRL методов машинного обучения: бустинга на основе деревьев принятия решений, метода опорных векторов, метода на основе нейронных сетей SNOW, классификаторов на основе максимальной энтропии, а также списков правил. Также авторы оценили результаты комбинирования этих классификаторов с помощью набора эвристик. Наилучшие результаты показала комбинация метода опорных векторов, максимальной энтропии и бустинга на основе деревьев принятия решений. Что касается индивидуальных классификаторов, авторы отмечают, что SVM лидирует по точности в ущерб полноте, а наиболее оптимальное сочетание точности и полноты достигается при использовании бустинга и деревьев принятия решений.

В последующий период были предприняты попытки как улучшить существующие результаты для английского языка, так и разработать системы автоматической классификации актантов для других языков. В ходе этих

исследований выяснилось, что синтаксис непосредственных составляющих недостаточно удобен для представления синтаксической информации в языках со свободным порядком слов и падежным маркированием. Было продемонстрировано, что **синтаксис деревьев зависимостей** в таких случаях обладает большей описательной силой [Johansson, Nugues, 2007; 2008].

Кроме того, было показано, что связь между задачами синтаксического и поверхностного семантического анализа — двусторонняя: не только автоматическая разметка актантов опирается на синтаксис, но и наоборот, синтаксический анализ может быть выполнен с лучшим качеством, если предоставить системе данные о семантических ролях. Один из первых подходов, в котором синтаксический и семантический анализ оказываются взаимозависимы, был предложен в 2008 году в работе [Haghighi, Toutanova, Manning, 2008]. Авторы использовали классификатор на основе максимальной энтропии со стандартным набором свойств на основе деревьев непосредственных составляющих, однако вместо единственного синтаксического представления классификация актантов выполнялась на ранжированном наборе синтаксических разборов, полученных автоматически. Для каждого из вариантов разбора производилась автоматическая классификация актантов, и затем выбирался разбор, для которого суммарная уверенность синтаксического парсера и SRL-компонента была максимальной.

Указанные выше тенденции привели к появлению нового типа систем, которые основывались на синтаксисе деревьев зависимостей. В 2007 и 2008 году были проведены соревнования CoNLL 2007 и 2008, посвящённые задаче автоматического синтаксического и семантического анализа как для английского, так и других языков. На нескольких работах, представленных на этих соревнованиях, мы хотели бы остановиться подробнее.

Исследование, описанное в работе Х. Льюис и Л. Маркес [Lluís, Màrquez, 2008] развивает идею, предложенную в [Haghighi, Toutanova, Manning, 2008].

Авторы предлагают систему, которая на основе тренировочных данных обучается одновременно выполнять синтаксический парсинг и автоматическую разметку актантов. Система состоит из пяти компонентов: предобработка, предварительный синтаксический анализ, идентификация предиката, финальный синтаксический и семантический парсинг и постобработка. На этапе предобработки из корпусных данных извлекаются стандартные для задач синтаксического анализа свойства. На этапе предварительного синтаксического анализа к данным применяется парсер с целью снабдить компонент классификации актантов синтаксическими свойствами. Затем в предложениях с помощью бинарного SVM-классификатора и набора эвристик выделяются целевые предикаты. После этого к данным снова применяется парсер, но на этот раз обученный с использованием комбинированных синтактико-семантических меток. В результате работы этого парсера для каждого предложения строится два дерева: синтаксическое и семантическое. Затем, на этапе постобработки, предикату приписывается значение, при котором роли оказываются максимально заполнены (своего рода снятие лексической неоднозначности, но выполненное не в начале анализа, а в конце). Система показала конкурентоспособные результаты по автоматической классификации актантов при стабильных результатах синтаксического анализа, однако авторы отмечают ухудшение качества при анализе именных предикатов.

Другая работа, о которой мы хотели бы рассказать в рамках нашего обзора – исследование Р. Йоханссон и П. Нюг [Johansson, Nugues, 2007], которые одними из первых стали использовать синтаксис зависимостей для автоматической разметки актантов, доказали состоятельность и продемонстрировали удобство этого подхода на примере работы с данными из корпуса FrameNet. Это одна из классических и наиболее цитируемых современных публикаций по автоматической разметке и классификации актантов. Авторы опираются на теоретический аппарат теории связывания

[Mel’čuk, 1988] и определяют задачу автоматической разметки актантов как задачу моделирования семантико-синтаксического интерфейса предиката. Для синтаксического анализа исходных предложений авторы обучили модель для парсера MaltParser [Nivre, Hall, Nilsson, 2006] на конвертированном в формат деревьев зависимостей корпусе Penn Treebank [Marcus, Santorini, Marcinkiewicz, 1993]. Далее они применили обученную модель к корпусу примеров FrameNet, в котором, в отличие от PropBank, отсутствует синтаксическая разметка. С помощью набора эвристик основанная на отрезках текста разметка была спроецирована на узлы деревьев зависимостей. Для снятия неоднозначности на уровне предикатов авторы использовали набор эвристик и классификатор на основе SVM, в результате чего каждому предикату из тестового корпуса приписывалось значение из FrameNet. После того как значение предиката определено, выполняется обнаружение и классификация актантов. В обоих случаях авторы используют классификатор на основе метода опорных векторов со стандартным набором свойств (отметим, что синтаксические свойства теперь извлекаются из деревьев зависимостей, а не составляющих). Роли приписываются актантам независимо, т.е. глобальной оптимизации не производится. Эта работа была предложена в рамках соревнования SemEval 2007, посвящённого автоматической классификации актантов на основе данных FrameNet, и продемонстрировала результаты, сопоставимые с системами на основе синтаксиса непосредственных составляющих. Авторы обращают внимание на ряд трудностей, связанных со свойством "путь" и на нетривиальность соответствий между синтаксическим и семантическим представлениями текста.

I.4 Современные системы

Современные системы автоматической классификации актантов опираются на более сложные методы, в которых информация о структуре задачи и особенностях семантического представления кодируется непосредственно в модели. В качестве примера такой системы можно привести систему SEMAFOR [Das и др., 2010]. Все рассмотренные нами ранее системы представляют собой последовательность независимых классификаторов: процесс обработки состоит из нескольких этапов, которые в зависимости от конкретной конфигурации могут включать в себя поиск целевого предиката, определение значения предиката (при использовании PropBank) или фрейма (при использовании FrameNet), а также обнаружение и классификацию актантов. Один из недостатков такого подхода — невозможность использовать информацию о присвоении ролей на этапе идентификации актантов. В системе SEMAFOR идентификация и классификация актантов выполняются одновременно. Система SEMAFOR получает на вход результаты предобработки исходных данных морфологическим анализатором, модулем снятия неоднозначности на основе WordNet [Fellbaum, 1998] и парсером деревьев зависимостей MST [McDonald, Lerman, Pereira, 2006]. Далее на основе извлечённых свойств происходит обучение двух классификаторов: одного для определения фрейма, и одного – для идентификации и классификации актантов. Обнаружение и классификация актантов, в отличие от более ранних работ, выполняются одновременно. Авторам удалось превзойти по качеству предыдущие разработки для классификации актантов на основе FrameNet. Система SEMAFOR до сих пор совершенствуется, подробный отчёт о работе системы, а также обзор конкурирующих подходов можно найти в работе [Das, 2010].

В рамках данного обзора мы рассматривали преимущественно системы, разработанные для английского языка. Как мы уже ранее упоминали, на сегодняшний день английский язык действительно наиболее разработан в релевантном для SRL отношении: доступно множество ресурсов для предобработки, а также корпуса, размеченные по семантическим ролям и предикатам. Для других языков, при наличии обучающих данных, используются подходы схожие с описанными выше, а качество автоматической классификации актантов, как правило, оказывается сопоставимым или несколько ниже, чем для английского языка. Основная сложность при разработке подобных систем состоит в зависимости от аннотированного ресурса: разметка по семантическим ролям – крайне трудозатратный и плохо формализуемый процесс, и даже наличие подобного корпуса для того или иного языка не гарантирует качественной работы обученных систем на новых данных. Эта проблема обычно известна как проблема **доменной специфичности** SRL.

Для решения проблемы доменной специфичности в области автоматической разметки семантических ролей в последние годы было предложено несколько подходов, которые позволяют уменьшить объём тренировочных данных, требуемый для обучения системы, или вовсе избавиться от необходимости в таких данных за счёт использования методов обучения без учителя.

Так, Х. Фюстенау и М. Лапата в работе [Furstenau, Lapata, 2011] предлагают технику **проекции аннотаций** с помощью выравнивания синтаксических графов на основе целочисленного линейного программирования. Общий принцип работы их системы состоит в следующем: корпус примеров FrameNet (*исходный корпус*) и некоторый другой большой, но не размеченный целевой корпус анализируются с помощью синтаксического парсера. Затем для каждого предложения из корпуса FrameNet в *целевом*

корпусе находится предложение-кандидат на проекцию. Эта операция может быть выполнена на основе леммы предиката или с использованием более сложного механизма снятия лексической неоднозначности. Далее синтаксические деревья исходного и целевого предложений фильтруются с помощью эвристик и подвергаются выравниванию на основании синтаксического и лексического сходства между узлами. Выравнивание производится с помощью целочисленного линейного программирования и имеет своей целью максимизировать сходство между графами. Поле того как выравнивание выполнено, семантическая разметка из исходного графа тривиальным образом переносится на целевой граф. Полученный расширенный набор данных (FrameNet плюс целевой корпус, обогащённый ролями) может использоваться для обучения системы.

В качестве примера системы обучения без учителя хотелось бы упонямуть решение, выполненное на основе графовой кластеризации, которое было предложено в работе [Lang, Lapata, 2011]. Авторы предлагают обработать исходный корпус, не содержащий семантической разметки, синтаксическим анализатором и расположить все актанты для каждого предиката на графе. Узлами графа будут являться употребления актантов в тексте, а рёбрами – отношения сходства между актантами, которые вычисляются на основе лексического и синтаксического сходства. К построенному таким образом графу прменяется алгоритм непараметрической графовой кластеризации Chinese Whispers [Biemann, 2006a], в результате работы которого граф оказывается разбит на группы сходных между собой узлов-актантов, которые и объявляются семантическими ролями для данного предиката.

Успех применения данного метода, как и многих других методов обучения без учителя применительно к высокоуровневым задачам, сильно зависит от качества предобработки корпуса и от моделей лексического сходства. В качестве альтернативного примера формулировки автоматической

разметки актантов как задачи непараметрической кластеризации можно упомянуть успешную работу [Titov, Klementiev, 2012], в которой для решения использовалась байесовская сеть со скрытыми переменными.

Как уже упоминалось, автоматическая разметка актантов – ресурсоёмкая задача, для решения которой требуется качественная предварительная обработка текстов и большие объёмы аннотированных данных. Несмотря на то, что наша система принадлежит к классическим системам обучения с учителем, кажется важным отметить потенциал современных методов, которые направлены на уменьшение зависимости SRL как от результатов предобработки, так и от объёмов тренировочных корпусов.

I.5 Автоматическая разметка актантов и русский язык

На сегодняшний день практически не имеется публикаций, посвященных решению задачи автоматической классификации актантов на русском материале.

Единственная известная нам реализация данной задачи на основе машинного обучения [Смирнов, Shelmanov, 2014] скорее относится к методам частичного обучения с учителем. Авторы опираются на реляционно-ситуационную модель текста [Осипов, Смирнов, Тихомиров, 2008]. В рамках этой модели актантами являются именные группы, а роль каждого участника определяется его семантическим классом, предложным оформлением и падежом. Ролевой инвентарь содержит около 60 семантических ролей абстрактного типа, например, "Агенс", "Пациенс" и т.д.

В рассматриваемой работе авторы описывают и оценивают две системы. Первая из них основана на правилах, и с помощью морфосинтаксических шаблонов способна распознавать семантические роли в новых текстах. На этапе предобработки исходное предложение обрабатывается морфологическим и синтаксическим анализатором, затем с помощью эвристических правил выбираются целевые предикаты и предположительные актанты, а затем с помощью правил и простых трансформаций актантам приписываются роли. На этапе постобработки производится оптимизация результатов и выбор значения предиката с помощью целочисленного линейного программирования.

Вторая система представляет больший интерес в контексте нашей задачи. Общая идея предложенного подхода состоит в следующем. С помощью тех же семантических правил предлагается разметить синтаксический корпус русского

языка СинTarPyc, а затем обучить синтаксический парсер MaltParser, используя набор смешанных семантико-синтаксических меток отношений (наподобие того как это было сделано в [Samuelsson и др., 2008]). Авторы приводят показатели качества для своей системы, однако оценивают это качество на корпусе, размеченном вручную, и только на предикатах, для которых представлены семантические правила, что отличается от стандартной процедуры оценки систем автоматической разметки актантов. Сравнить качество подхода, основанного на правилах, и подхода, основанного на машинном обучении, в этом контексте не представляется возможным, т.к. семантико-синтаксический парсер опирается на правила. Тем не менее, нам кажется, что предложенная технология имеет хорошие перспективы, в том числе с точки зрения практического применения. В то же время, кажется затруднительным сравнить результаты, полученные в данной работе, с нашими из-за разницы в подходах и в исходном материале.

Для русского языка также существует несколько систем на основе правил (например, [Anisimovich и др., 2012] и [Ермаков, Плешко, 2009]), а также систем извлечения фактов, в том числе позволяющих генерировать шаблоны описания событий на основе больших массивов неразмеченных данных. Так, в работе [Котельников, Лукашевич, 2012] описан метод извлечения шаблонов из текстов на основе целевых слов и примера заполнения участников ситуации. Система находит описания выбранной ситуации в текстах и на основе примеров реализации конкретных участников события выделяет шаблоны, которые затем могут быть использованы для извлечения информации о новых ситуациях. Подобные системы имеют высокую практическую ценность, однако решаемая в рамках этих исследований задача отличается от нашей, т. к. системы не опираются на лексикографические ресурсы и не имеют своей целью извлечение семантических ролей в лингвистическом понимании.

Существует по крайней мере две причины, по которым автоматическая классификация актантов практически не разрабатывается для русского языка в том виде, в котором она реализуется в западной компьютерной лингвистике.

Во-первых, автоматическая классификация актантов – ресурсоёмкая процедура в плане требований к предварительной обработке документов. Как уже было сказано выше, синтаксическая и лексико-семантическая информация очень важна для автоматической классификации актантов. Поскольку автоматическая классификация актантов находится в самом конце цепочки обработки текста, она аккумулирует в себе ошибки всех предыдущих этапов, начиная от разделения текста на слова и заканчивая синтаксическим анализом и анализом анафоры. В последние годы инструменты для предобработки текстов для русского языка активно развиваются. Так, в рамках конференции "Диалог" в 2010 году состоялось соревнование морфологических анализаторов [Ляшевская и др., 2010], в 2012 году прошло соревнование синтаксических парсеров [Толдова и др., 2012], в 2014 году – соревнование модулей разрешения анафорической неоднозначности [Toldova и др., 2014], а в 2015 году прошло соревнование систем по определению семантической близости слов [Panchenko и др., 2015]. Отрасль активно развивается, и можно надеяться, что в ближайшее время в руках исследователей окажутся все необходимые инструменты для семантического анализа текстов. В то же время следует отметить, что отдельной задачей является обеспечение совместимости между результатами работы различных морфологических и синтаксических анализаторов. По причине отсутствия общепринятого стандарта для входных и выходных данных (как с точки зрения формата, так и с концептуальной точки зрения), зачастую оказывается проблематичным сконструировать полную цепочку предварительной обработки для решения той или иной задачи. Существуют примеры подобной унификации для английского и немецкого

языков [Castilho de, Gurevych, 2014], и мы надеемся, что в ближайшем будущем эта проблема будет решена и для русского языка.

Вторая причина, по которой, на наш взгляд, рассматриваемая задача не пользуется большой популярностью на русском материале, состоит в отсутствии подходящего ресурса, на котором могло бы быть выполнено машинное обучение. На данный момент в разработке находится ресурс FrameBank [Lyashevskaya, Kashkin 2015], который предоставляет корпус конструкций, размеченный по семантическим ролям. В последующих главах нашей работы мы остановимся на этом ресурсе более подробно, т.к. именно он используется в качестве основы для обучения нашей системы. Учитыая, что для автоматической разметки актантов объём тренировочных данных имеет решающее значение, и что расширение подобного ресурса – трудоёмкая задача, кажется перспективным опробовать методы частичного обучения с учителем, разработанные для английского языка, для автоматического расширения корпуса FrameBank. Например, одним из направлений, которое могло бы значительно ускорить разработку ресурса FrameBank и расширить степень его применимости, могла бы стать упомянутая выше проекция аннотаций как на внешние неразмеченные корпуса, так и на синтаксический корпус СинТагРус. Можно рассматривать это как комбинацию подходов, предложенных [Смирнов, Shelmanov, 2014] и [Котельников, Лукашевич, 2012], в которой правила для извлечения примера выводятся на основе обучающего корпуса FrameBank или аналогичного ресурса. Также кажется перспективным использование методов машинного обучения без учителя, однако в данном случае мы попадаем в зависимость от качества предобработки, и пока к применению этих подходов на русском материале, как нам кажется, следует относиться с осторожностью.

Выше мы постарались поместить наше исследование в теоретический и прикладной контекст и указать на наиболее важные и интересные, с нашей

точки зрения, аспекты автоматической классификации актантов. Система автоматического выделения семантических ролей, разработанная в рамках диссертационного исследования, является первой системой для русского языка, основанной на ресурсе FrameBank. Следующие главы диссертации будут посвящены описанию и обоснованию моделей, использованных для разработки системы, а также процедуре оценки ее работы и анализу и полученных с её помощью результатов.

II. Система автоматической разметки актантов для русского языка

II.1 Постановка задачи

Существует несколько вариантов постановки задачи автоматической классификации актантов и оценки качества результирующей системы. Помимо разделения на системы, работающие с высокоуровневым инвентарём FrameNet, и системы, основанные на предикатно-специфических ролях, наподобие PropBank, существует также несколько важных параметров, которые, не будучи чётко артикулированными, затрудняют понимание принципов работы системы и интерпретацию полученных результатов. В этой главе мы опишем наиболее распространённые варианты формулировки задачи SRL, а также формально специфицируем ту задачу, которая будет решаться в рамках настоящего исследования.

Итак, первое и наиболее важное концептуальное разделение между системами автоматической классификации актантов – это разделение на системы, оперирующие абстрактными **высокоуровневыми ролями**, т.е.

ролями из фиксированного абстрактного инвентаря наподобие FrameNet и Verbnet [Das, 2010; Gildea, Jurafsky, 2000], и системы, выделяющие **предикатно-специфические роли**, где роли уникальны для каждого предиката [Johansson, Nugues, 2008]. Для английского языка в качестве исходных данных в первом случае используется система и корпус FrameNet, в котором семантические роли организованы в иерархию, и разбор предложения, выполненный системой SRL, может выглядеть следующим образом:

[Abby]$_{Buyer}$ bought [a car]$_{Goods}$ [from Robin]$_{Seller}$ for [$5000]$_{Money}$.

Пример 8: Разбор предложения в формализме FrameNet

Системы второго типа, напротив, приписывают семантическим аргументам конкретные, предикатно-специфические роли, и основываются, как правило, на корпусе PropBank. В этом случае результат обработки предложения может выглядеть так:

[Abby]$_{Arg0}$ **bought** [a car]$_{Arg1}$ [from Robin]$_{Arg2}$ for [$5000]$_{Arg3}$.

Пример 9: Разбор предложения в формализме PropBank

Как правило, при разработке ресурсов для других языков создатели опираются на опыт либо FrameNet, либо PropBank, и результирующие ресурсы по своим свойствам оказываются похожи на свой англоязычный прототип. Каждый из подходов имеет свои преимущества и недостатки с точки зрения автоматической классификации актантов.

Так, абстрактные роли FrameNet могут быть использованы для описания нескольких предикатов, а актанты, маркированные этими ролями, часто имеют общие семантические свойства, что упрощает присвоение семантической роли

вне зависимости от выбранного метода автоматической классификации. Для наглядности рассмотрим следующий синтетический пример.

[Петя]$_{Покупатель}$ **купил** [машину]$_{Товар}$ → Вася **продал** машину

Пример 10: Общие роли в рамках фрейма

Благодаря тому, что системе заранее известно, что предикаты "купить" и "продать" относятся к одному и тому же фрейму и разделяют один набор актантов, на этапе автоматического анализа система будет принимать решение не о том, соответствует ли в лексическом отношении тот или иной актант типичному заполнителю выбранной роли для выбранного предиката, а о том, насколько вероятно, что данный актант выражает выбранную абстрактную семантическую роль, которая представлена для нескольких предикатов. Это полезное обобщение помогает системе принимать решение в случаях, когда исходный ресурс содержит мало примеров для одного предиката, но много примеров для другого, принадлежащего к тому же фрейму, что и первый.

В то же время использование высокоуровневых ролей может привести к сложностям из-за избирательной регистрации в ресурсе тонких семантических различий между актантами и фреймами. Учитывая, что основная цель разработчиков FrameNet – описательная, зачастую различия между ролями, будучи важными с позиций теоретической лингвистики, оказываются слишком специфическими с точки зрения прикладных задач автоматической обработки языка. Более того, эти различия не всегда проводятся системно, что затрудняет автоматический анализ подобных ресурсов. Пожалуй, любой, кто принимал участие в непосредственной разметке текстов на основе ресурса типа FrameNet, согласится, что некоторые случаи представляют трудности даже для людей, и, несомненно, эта задача оказывается ещё более сложной для машин.

В случае с FrameNet (и в отличие от VerbNet) дополнительная сложность состоит в том, что выбор конкретной метки актанта зачастую зависит от выбранного фрейма: так, абстрактный Агенс при предикате "купить" получит роль "Покупатель", а при предикате "убить" – роль "Убийца". Таким образом, автоматическая система должна не только выделять роли, но и определять фрейм для выбранного предиката, что не всегда является тривиальной задачей. Поскольку разметка примеров FrameNet имеет в первую очередь иллюстративную цель, разработчики ресурса заинтересованы в том, чтобы описать как можно больше фреймов и продемонстрировать различия между ними. В результате этого задача автоматической разметки актантов с использованием ролей FrameNet усложняется.

PropBank и подобные ему ресурсы лишены этого недостатка. В них роли размечаются для каждого предиката независимо от его лексического значения, при этом дополнительно существует общий набор модификаторов, семантика которых однозначна и легко поддаётся интерпретации. Теряя способность к семантическим обобщениям, системы на основе PropBank получают взамен большую строгость исходного материала. Кроме того, подобные системы лучше поддаются оценке, т.к. различия между предикатно-специфичными ролями более очевидны для эксперта и разметчика тестового материала.

Вопрос о предпочтении того или иного подхода не имеет однозначного ответа, и конечный выбор зависит в большой степени от контекста, в котором планируется проводить исследование или использовать разработанную систему, а также от того, какие ресурсы доступны в момент исследования для целевого языка.

К вопросу выбора между FrameNet-подобной и PropBank-подобной парадигмой примыкает вопрос о том, каким образом осуществляется работа с неизвестными предикатами, которые не описаны в исходном ресурсе. В случае когда система разрабатывается на основе абстрактных ролей, у нас есть

возможность предсказывать роли актантов даже в отсутствие предиката путём построения модели для каждой отдельной роли. Например, система может научиться определять роль "*Инструмент*" или роль "*Товар*" на основе типичных свойств актантов, заполняющих эту роль. В то же время следует понимать, что возможности генерализации в данном случае ограничены, т.к. некоторые различия выражаются только на уровне синтаксиса, например, для ролей "*Покупатель*" и "*Продавец*" лексическое наполнение может быть очень сходным. В случае же конкретных, предикатно-специфических ролей работа с неизвестными предикатами затрудняется (однако модификаторы, общие для всех предикатов могут быть успешно определены). Если абстрактные роли позволяют нам надеяться на некоторое "постоянство" в присвоении ролей актантам неизвестного предиката (так, например, мы можем рассчитывать, что семантика роли "Агенс" будет сходной для различных предикатов), то в случае в конкретными ролями такой гарантии мы не имеем (роль *Arg0* может иметь самые разные соответствия на уровне абстрактных ролей).

Следующий важный момент, определяющий специфику постановки задачи – включается ли определение конкретного значения предиката в задачу автоматической разметки актантов или же предполагается, что его значение дано изначально и поступает из модуля предобработки. Несмотря на то, что существуют работы, выполняющие помимо классификации актантов также определение значения предиката, в большинстве исследований эта задача справедливо относится к области снятия лексической/глагольной неоднозначности и подробно не рассматривается. Для того чтобы система могла быть применена на практике, неоднозначность должна быть снята, поэтому в промышленных системах используются готовые компоненты и техники для снятия неоднозначности. В исследовательских же системах зачастую принимается такая установка, при которой конкрентные значения предикатов даны системе изначально, что позволяет сконцентрировать усилия

на разработке решения задачи автоматического выделения ролей и, что существенно, впоследствии оценивать качество работы именно SRL-компонента.

Формулировка задачи зависит и от того, что именно является объектом классификации. Решение в данном случае определяется тем, какой именно элемент разбираемого предложения мы считаем носителем семантической роли. Как правило, в современных системах носителем роли объявляется составляющая или узел дерева зависимостей. В то же время, в корпусах, сопутствующих SRL-ресурсам, зачастую используется разметка по отрезкам текста, как в следующем примере.

Newspapers reporting a briefing by parliament speaker Ahmed alSaadoum today, did not **NAME** the official involved of his function.

Пример 11: Несоответствие разметки синтаксической структуре (FrameNet)

Как можно видеть из этого примера, используемая в данном случае разметка не соответствует ни делению на зависимости, ни делению на составляющие. Подобное решение мотивируется нежеланием привязывать разметку по семантическим ролям к тому или иному синтаксическому формализму, однако приводит к дополнительным трудностям при автоматической классификации актантов. Так, в случаях, когда оценка качества системы производится автоматически на основе тестовой выборки, система должна не только выполнять свою прямую задачу, но и успешно "имитировать" стиль разметки, принятый при аннотации выбранного ресурса. Автоматическая разметка актантов может быть сформулирована как задача **разметки последовательностей**, в этом случае система должна для каждого актанта определить его начало и конец, а затем приписать этому актанту роль. Однако

более естественным способом формулировки задачи SRL в последние годы считается **разметка на основе узлов в дереве зависимостей**, при которой классификации подвергается только один объект, а именно, главный узел поддерева, соответствующего выбранному аргументу. Объединённые системы, которые производят синтаксический и семантический анализ одновременно (например, [Lluís, Carreras, Màrquez, 2013]), также оперируют только одним словом-узлом, а результат их работы – дерево, в котором узлы связаны семантическими и синтаксическими отношениями. Тем не менее, в случае, когда задача сформулирована в терминах узлов, а не отрезков, возникает проблема поиска соответствия между отрезком в исходной разметке и узлом синтаксического дерева. Как правило, эта проблема решается с помощью эвристических правил.

Наконец, в архитектурном плане системы можно разделить на два класса в зависимости от того, каким образом выполняется классификация актантов. Системы первого типа сначала определяют, является ли объект классификации семантическим аргументом выбранного предиката, а затем приписывают ему конкретную роль. Системы второго типа выполняют эти операции одновременно.

Системы также различаются в зависимости от того, производится ли **глобальная оптимизация** результатов классификации, т. е., влияют ли решения классификатора относительно слов в предложении друг на друга, или же разметка для каждого слова и каждого актанта выполняется независимо.

Итак, мы рассмотрели основные параметры, по которым системы автоматической обработки актантов различаются между собой. Рассмотренная классификация позволяет более четко сформулировать задачу автоматической разметки актантов в том виде, в котором она будет решаться в рамках данного исследования.

Мы интерпретируем SRL как задачу классификации на основе предикатно-специфических ролей. В общем виде задача классификации формулируется следующим образом. Необходимо построить решающую функцию вида $f(x) = y$, где x – объект классификации или экземпляр, а y – метка класса. Функция строится на основе тренировочной выборки T, состоящей из набора экземпляров, класс которых известен, с помощью алгоритма, задача которого – подобрать функцию из заранее заданного семейства функций, которая наиболее точно описывала бы тренировочные данные. Экземпляры описываются в терминах свойств или признаков $(a_1, a_2 \dots a_n)$, и конечная задача обучения – построить функцию $f(a_1, a_2, \dots a_n) = y'$, которая правильно предсказывает класс для новых, ранее не представленных классификатору объектов. Множество доступных в рамках выбранной задачи признаков формирует **признаковое пространство**. В качестве простой иллюстрации задачи классификации приведём пример, в котором объекты – это точки в признаковом пространстве координат, а класс – это цвет точки. Обучающая выборка представлена на Рис. 2:

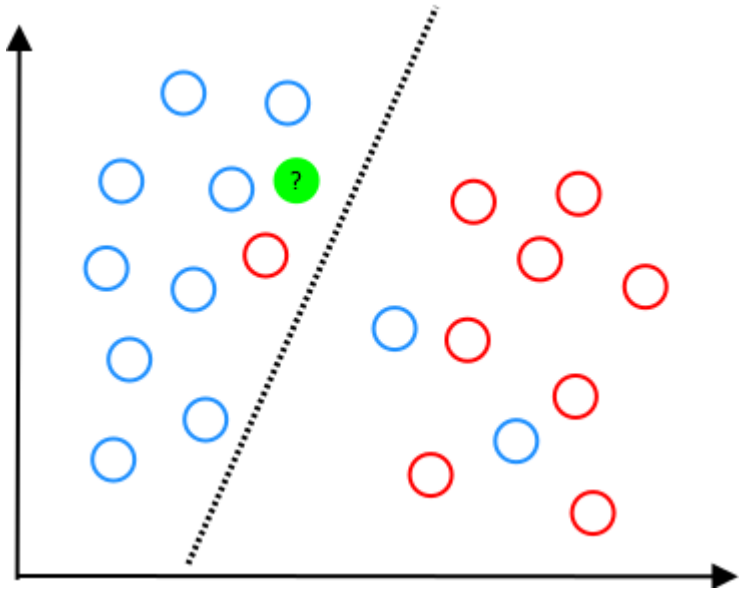

Рисунок 2: Задача классификации в двумерном пространстве

В процессе обучения строится решающая функция f (представленная на рисунке линией), которая делит признаковое пространство на части. Затем, когда система получает на вход новый экземпляр, эта функция используется для того, чтобы определить, к какому классу этот экземпляр принадлежит. Как правило, для оценки систем машинного обучения исходные данные, снабжённые метками классов, случайным образом разбиваются на непересекающиеся **тренировочную** и **тестовую** выборки. Тренировочная выборка используется для обучения модели. Тестовая выборка содержит новые экземпляры, ранее не представленные классификатору (однако содержащие "правильные" метки классов). Применив обученный классификатор к тестовой выборке, можно затем сравнить результат работы классификатора с реальными значениями классов и оценить качество работы классификатора.

В нашем случае объектом классификации, или экземпляром, мы будем **считать узел в дереве зависимостей**. Нам показалось разумным остановить выбор на "узловой" интерпретации как более разработанной в современной литературе и более подходящей для русского языка с учётом доступных ресурсов предобработки. В соответствии с формализмом деревьев зависимостей, каждому узлу соответствует одно слово предложения. Каждый узел определяется в терминах набора его свойств и имеет метку класса, которая соответствует семантической роли актанта, выражаемого этим узел. Узлы, не выражающие актантов, получают специальную отрицательную метку класса *None*. Задача классификатора состоит в том, чтобы на основании обучающей выборки построить модель, которая позволяет, зная свойства узла, предсказать его класс, т.е. семантическую роль. К результатам работы классификатора затем применяется модуль глобальной оптимизации на основе целочисленного программирования.

Обучение классификатора производится на тренировочной выборке, которая состоит из случайно отобранных и определенным образом сгруппированных предложений исходного корпуса примеров FrameBank (о принципах группировки будет подробно рассказано ниже). Оценка качества работы классификатора производится на тестовой выборке, которая также представляет собой набор предложений исходного корпуса.

Для того чтобы сконцентрироваться на задаче автоматической разметки актантов, мы приняли решение не включать модуль снятия глагольной неоднозначности в нашу систему и исходим из того, что значения глаголов даны нам заранее.

Наконец, мы производим поиск и классификацию актантов в один шаг, т.к. в формулировке на основе узлов дерева зависимостей необходимость в двухступенчатой архитектуре классификатора неочевидна.

Теперь, когда мы сформулировали задачу, которую предстоит решить, рассмотрим подробнее ресурс, на основе которого будет производиться обучение и тестирование классификатора, – корпус FrameBank – а затем перейдём к описанию реализации системы.

II.2 Исходные данные

В рамках данного исследования задача автоматической классификации актантов формулируется как задача машинного обучения с учителем. Для того чтобы решить её, нам необходимы тренировочные данные, а именно, корпус текстов, в котором были бы размечены предикаты и их значения, а также сами семантические роли. В качестве такого корпуса мы использовали коллекцию примеров из корпуса FrameBank. Ниже мы остановимся на этой системе подробнее.

FrameBank представляет собой корпусно-лексикографический ресурс, описывающий лексические конструкции русского языка с помощью специальным образом размеченных предложений из Национального корпуса русского языка [Апресян, Богуславский, Иомдин, 2005].

На сегодняшний день ресурс находится в стадии активной разработки: на данный момент в корпусе представлены примеры для ок. 2200 лексем (в основном глагольных), для каждой лексемы в среднем размечено по 100 примеров. Этих данных (с некоторыми оговорками) уже достаточно для использования корпуса в исследовательских задачах.

Центральным организующим компонентом системы разметки, используемой в FrameBank, является лексическая конструкция. Лексические конструкции в семантическом отношении соответствуют значениям предикатов. Каждый глагол может иметь (и, как правило, имеет) несколько конструкций. Описание конструкций в системе FrameBank представляет каждую конструкцию в виде шаблона, для которого указываются следующие характеристики:

- уникальное имя конструкции
- состав элементов конструкции

- морфологические и синтаксические свойства элементов
- экспликация семантической роли участника
- семантические ограничения на участников конструкции

В качестве примера приведём описание конструкции *купить_1.1*:

Паспорт конструкции: ***1.1_Пойди купи хлеба, молока и яиц.***

Схема: Snom V {Sacc / Spart}
Пример: Пойди купи хлеба, молока и яиц. Ко дню рождения мама купила мне платье [отрез на костюм]. Он купил недавно небольшую яхту [кооперативную квартиру, машину].

Буква	Вершина	Экспликация	Ранг	Сем. ограничения	Заполнение вершины	Заполнение группы
X	Snom	конечный посессор	Субъект	лицо		
-	купить	-	Предикат	-		
W	{Sacc / Spart}	пациенс	Периферия	-		

Рисунок 3: Паспорт конструкции в системе FrameBank

Для конструкции *купить_1.1* задаётся множество из двух актантов, каждый из которых получает букву-идентификатор (*X* и *W*). Для каждого актанта указывается стандартный способ реализации, даётся подробная интерпретация его семантики (столбец “*Экспликация*”), указывается синтаксический ранг и семантические ограничения. Также для каждой конструкции, помимо примеров из корпуса, приводится несколько канонических примеров употребления (в нашем примере “*Пойди купи хлеба, молока и яиц*” и проч.).

Как мы можем видеть из этого примера, с точки зрения теории семантических ролей FrameBank использует гибридное кодирование: каждый участник конструкции получает уникальную специфичную для конструкции роль (обозначенную буквой), а также экспликацию в форме семантической пометы. Инвентарь ролей, используемых в качестве семантических помет,

организован иерархически и содержит порядка 88 ролей, начиная от общих ("*Агенс*", "*Пациенс*") и заканчивая частными в случаях, когда общих ролей недостаточно для описания семантики участника ("тот, кому служат"). Допускается использование сдвоенных ролей и расщепления ролей. Отдельный блок ролей отвечает за типичные модификаторы-сирконстанты ("*Время*", "*Место*" и т.д.) подобно тому, как это сделано в системе PropBank. Поскольку проект находится в стадии разработки, инвентарь пополняется новыми ролями.

Каждой конструкции в системе FrameBank соответствует набор примеров из Национального корпуса русского языка (НКРЯ). Примеры представляют собой отрывки текста, разбитые на предложения и слова. Для каждого слова дана морфологическая информация, полученная с помощью автоматического анализатора, а также семантические пометы из инвентаря НКРЯ. В ходе разметки аннотаторы соотносят каждый предикат предложения с соответствующей лексической конструкцией. Далее они отмечают отрезки текста, которые, по их мнению, относятся к той или иной роли. Разметка производится с помощью предикатно-специфических ролей (буквенных идентификаторов); комбинация имени конструкции и специфичной роли позволяет однозначно определить остальные характеристики выбранного участника на основе словаря конструкций. Разметка производится в первую очередь для глагольных конструкций, включая конструкции с нефинитными формами (причастиями, деепричастиями, инфинитивами и т.д.), что увеличивает сложность распознавания семантических ролей в рамках одной конструкции.

Следующий пример демонстрирует разметку, которая используется в качестве исходных данных для нашей системы.

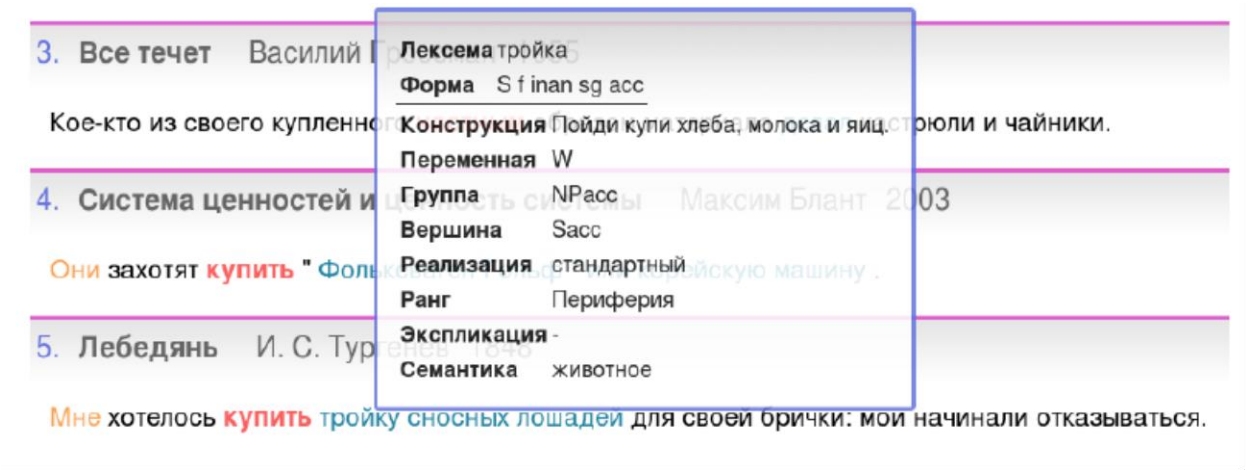

Рисунок 4: Разметка в корпусе FrameBank

С концептуальной точки зрения FrameBank занимает промежуточное положение между PropBank и FrameNet и учитывает опыт разработки и использования этих ресурсов. С PropBank выбранную нами систему роднит использование специфичных ролей и выделение модификаторов в отдельный класс. Сходство с FrameNet в первую очередь обусловлено использованием иерархической системы ролей, отсутствием синтаксической разметки в корпусе (этот аспект имеет практическую важность) и группировка описательных единиц в семантическую сеть. В то же время, в отличие от FrameNet, система FrameBank опирается не на понятие фрейма, а на понятие конструкции, мотивируя это тем, что "конструкция каждого предиката имеет индивидуальные особенности, даже если они относятся к одному фрейму" [Ляшевская, Кашкин, 2013]. Можно сказать, что FrameNet более "семантичен" и ориентируется в первую очередь на фреймовую семантику [Fillmore, 1982], в то время как FrameBank описывает явления более поверхностного уровня и опирается на грамматику конструкций [Goldberg, 1995; Рахилина, 2010] и теоретические и прикладные исследования Московской семантической школы [Апресян и др., 2010]. FrameNet практически не ограничивает тенденцию к дроблению ролей, в результате чего ролевой инвентарь оказывается практически бесконечным, и это создаёт определенные трудности как в

процессе разметки, так и при использовании этого ресурса в качестве источника данных при разработке приложений. Учитывая этот опыт, FrameBank по мере возможностей поддерживает инвентарь ролей небольшим, при этом сохраняя предикатно-специфическое маркирование.

Система FrameBank на момент начала исследования находилась на стадии разработки, и описания конструкций, а также некоторые корпусные примеры, содержали неточности. В связи с этим было принято решение использовать лишь крайне небольшой, однако наиболее стабильный и надёжный фрагмент доступной разметки, а именно разметку по именам конструкций и специфичным ролям. Учитывая рамки поставленной задачи (система не работает с неизвестными предикатами и не моделирует семантические роли независимо от предиката), данных о разметке по специфичным ролям для выполнения нашей задачи оказывается достаточно. Полученный набор данных имеет сходство с корпусом PropBank, однако в отличие от последнего не содержит синтаксической разметки. Для того, чтобы добавить этот важный для автоматической классификации актантов уровень представления, мы разобрали корпус FrameBank с помощью синтаксического анализатора. Поскольку аннотация FrameBank выполнялась по отрезкам текста, мы также выполяем проекцию аннотаций с отрезков текста на узлы дерева зависимостей. В результате этих манипуляций корпус получает синтаксическую разметку и может быть использован для обучения системы автоматической классификации актантов. Ниже мы рассмотрим эти компоненты, а также другие операции, которые мы производим над исходными данными.

II.3 Описание системы

II.3.1 Основные компоненты системы

Начать описание разработанной нами системы кажется уместным с представления её общей архитектуры. Ее можно условно разделить на следующие модули: модуль препроцессинга (фильтрация, морфологический анализ, лемматизация, синтаксический анализ), модуль обогащения данных (проекция на узлы), модуль обучения (извлечение свойств, классификатор, ILP-оптимизация). Приведённая ниже схема (Рис. 5) иллюстрирует взаимодействие модулей системы.

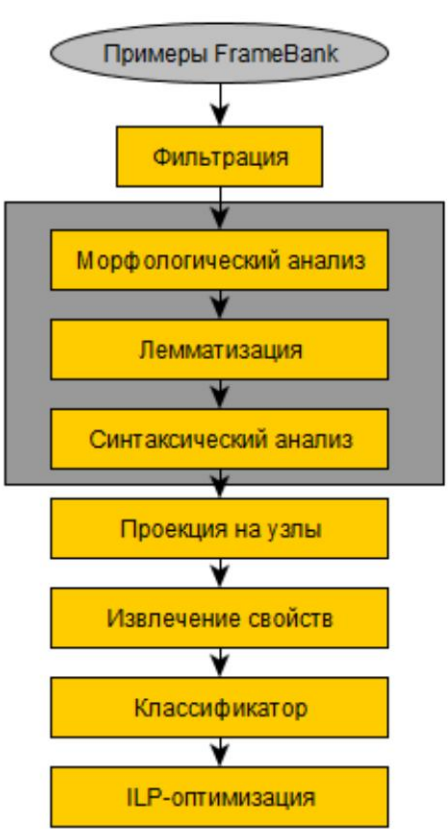

Рисунок 5: Архитектура системы автоматической разметки актантов

Итак, на вход системе поступает база данных FrameBank, которая помимо прочего содержит размеченные по семантическим ролям примеры употребления конструкций из Национального Корпуса русского языка в формате xml. Поскольку ресурс находится на стадии разработки, некоторые примеры в корпусе содержат ошибки разметки, связанные в большинстве случаев с техническими причинами. Для того чтобы дальнейшая работа была возможной, мы применяем процедуру фильтрации корпуса, в результате которой на основании простых правил принимаем решение, какие из предложений будут использованы в эксперименте. К примерам из корпуса были применены следующие фильтры:

- Пример должен содержать предикат
- Пример должен начинаться с заглавной буквы или символа и заканчиваться знаком препинания
- Пример должен представлять собой одно полное предложение

Мы считаем, что подобная фильтрация не оказывает значительного влияния на задачу и не отдаёт предпочтения тем или иным предикатам и конструкциям в ущерб другим. По мере развития корпуса FrameBank необходимость в этом этапе, как мы надеемся, исчезнет.

Предложения корпуса FrameBank разбиты на слова и содержат слой морфологической разметки. Синтаксические разборы предложений отсутствуют. Учитывая, что слой морфологической разметки создан автоматически и содержит морфологическую неоднозначность, мы приняли решение удалить всю информацию из корпуса кроме непосредственно разбиения на слова и предложения и разметки по семантическим ролям. После этого корпус был обработан морфологическим анализатором со снятием неоднозначности и синтаксическим парсером из пакета, разработанного С. Шаровым [Sharoff, Nivre, 2011]. Принципильно важно на этом этапе предвидеть возможные ошибки, которые возникают на этапе работы парсера. Так, в редких

случаях в процессе предобработки возникают границы предложений, которые отсутствовали в исходных данных. В таких случаях приоритет отдаётся границам, предложенным морфологическим анализатором и парсером, для сохранения синтаксических разборов. Результатом работы модуля предобработки является файл в формате CoNLL-2009 [Hajič и др., 2009], который обогащается ролями из разметки FrameBank.

Как уже ранее упоминалось, разметка во FrameBank выполнена по отрезкам текста, а не по синтаксическим узлам или группам. Учитывая, что мы определили задачу автоматической классификации актантов как задачу разметки узлов, нам необходимо осуществить отображение разметки FrameBank с отрезков текста на узлы соответствующего синтаксического дерева. За эту операцию отвечает модуль обогащения данных, о котором мы ещё расскажем позднее. Этот этап завершает процесс предварительной обработки данных. Ниже приводится пример предобработки, на котором отражены различные стадии этого процесса. Колонки соответствуют стадиям предобработки и добавляются последовательно, слева направо.

I	II	III	IV		V		VI		VII
Индекс	Токенизация	Лемматизация	Часть речи	Морфология	Вершина	Имя отношения	SRL отрезки	SRL информация	SRL узел
1	Второй	второй	M	Momsn	2	опред	X	X	_
2	ученик	ученик	N	Ncmsny	3	предик	X	X	волноваться 1.1#X
3	говорил	говорить	V	Vmis-sma-e	0	ROOT			
4	,	,	,	,	3	PUNC			
5	судорожно	судорожно	R	R	6	обст			
6	волнуясь	волноваться	V	Vmgp---m-e	3	обст	PRED	волноваться 1.1	волноваться 1.1#@PRED
7	,	,	,	,	6	PUNC			
8	тиская	тискать	V	Vmgp---a-e	6	обст			
9	в	в	S	Sp-l	8	обст			
10	руках	рука	N	Ncfpln	9	предл			
11	какую-то	какую-то	P	P--fsaa	12	опред			
12	рукопись	рукопись	N	Ncfsan	8	1-компл			

Рисунок 6: Этапы предварительной обработки (слева направо)

Далее все предложения-примеры из корпуса группируются по конструкциям, которые они описывают, формируя таким образом подкорпуса примеров для каждой отдельной конструкции.

Для каждого из полученных подкорпусов производится случайное разбиение на тренировочную и тестовую выборки. Единицей разбиения мы принимаем предложение (а не слово, что, впрочем, было бы вполне корректно учитывая нашу "пословную" постановку задачи). Тренировочная выборка поступает на вход классификатора и используется для обучения, тестовая выборка используется для оценки качества работы классификатора.

Как тренировочная, так и тестовая выборка поступают на вход модуля извлечения свойств, который преобразует информацию, полученную в результате предварительной обработки, в свойства, используемые классификатором. Модуль извлечения свойств приписывает набор признаков каждому узлу дерева зависимостей, построенного для каждого предложения тренировочной и тестовой выборки. Здесь же экземпляры-узлы получают метку класса: в тренировочной выборке эта метка используется для обучения классификатора, а в тестовой – для сравнения результатов работы системы с эталонной разметкой.

На этапе тестирования каждое предложение тестовой выборки подаётся на вход классификатору, который для каждого узла в дереве зависимостей этого предложения определяет его семантическую роль. Ключевая проблема, которая возникает на данном этапе состоит в том, что одна и та же роль может быть приписана нескольким узлам. Такой результат противоречит базовым принципам теории семантических ролей, в соответствии с которой все семантические роли актантов должны быть уникальны. Это имеет неприятные последствия и с практической точки зрения, поскольку не позволяет однозначно определить роли актантов, а это значит, что последующее использование результатов работы системы в других прикладных задачах будет осложнено. Для решения этой проблемы было решено использовать модуль ILP-оптимизации на основе метода целочисленного программирования для постобработки результатов классификации. Задача модуля – для каждого

предложения выбрать наилучшую комбинацию решений классификатора, которая удовлетворяет требованию, чтобы каждая роль была приписана только один раз. Подробнее работа модуля будет рассмотрена ниже. Результат работы модуля оптимизации является конечным результатом работы системы и поступает на выход.

Далее будут рассмотрены те модули системы, которые были специально разработаны в рамках текущего исследования и которые, собственно, и составляют научную ценность и новизну предлагаемой системы. Это во-первых модуль проекции на синтаксические узлы, во-вторых, модуль отбора свойств, в-третьих, собственно модуль классификатора, и в-четвертых, модуль ILP-оптимизации.

II.3.2 Модуль проекции на синтаксические узлы

Предназначение описываемого модуля состоит в том, чтобы сопоставить узлы синтаксического дерева с разметкой по семантическим ролям. Поскольку единого стандарта разметки текстов по семантическим ролям не существует, различные ресурсы используют разные конвенции аннотирования аргументов. В частности, английский FrameNet и русский FrameBank используют схему разметки, при которой границы аргументов задаются отрезками текста, а не синтаксическими узлами. Это создает определенные сложности при определении того, какие именно узлы синтаксического дерева являются представителями той или иной семантической роли. В качестве иллюстрации данной проблемы рассмотрим следующий пример:

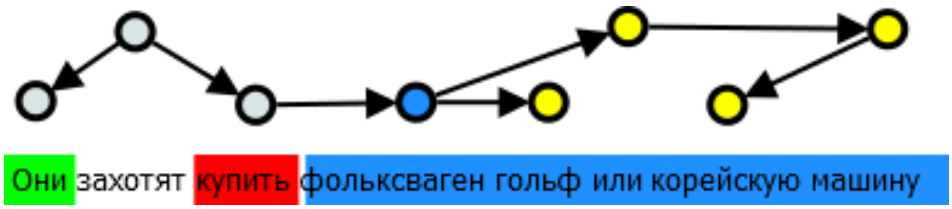

Рисунок 7: Проекция разметки на синтаксические узлы

В данном случае сложность представляет аргумент "*Фольксваген Гольф или корейскую машину*", который включает в себя пять синтаксических узлов. Для извлечения свойств для классификации нам нужно определить "главный" представительный узел для выбранного отрезка, и далее считывать свойства только этого узла.

Как правило, для решения этой проблемы используется набор эвристик, основаннных на максимизации пересечения **покрытия** зависимостного узла с аннотацией [Bauer, Fürstenau, Rambow, 2012]. Под покрытием узла понимается набор всех его зависимых, в том числе и непрямых. Тот узел, покрытие которого максимально пересекается с аннотацией, объявляется представителем этой аннотации в дереве зависимостей. Так, приведенный выше проблематичный случай мог бы быть разрешен следующим образом. Рассмотрим покрытие для всех узлов, которые расположены в рамках отрезка, обозначенного аннотацией.

узел	**покрытие**
Фольксваген	Гольф, или, машину, корейскую
Гольф	-
или	машину, корейскую
машину	корейскую
корейскую	-

Таблица 1: Покрытие узлов синтаксического дерева

Покрытие узла *Гольф* включает в себя только само слово “Гольф”, пересечение с исходной аннотацией составляет 5 из 35 символов или около 12%. Покрытие узла *или* пересекается с аннотацией в 15 из 35 символов. Наконец, узел *Фольксваген* имеет наибольшее пересечение с исходной аннотацией, и потому выбирается в качестве ролевого узла для данного предиката.

В результате применения описанной выше процедуры каждой семантической роли ставится в соответствие узел дерева зависимостей. После этого каждый узел дерева преобразуется в экземпляр: объект, описанный в терминах свойств и наделённый меткой класса. Эти экземпляры используются для обучения классификатора.

II.3.3 Модуль классификатора

После того как узлы входных синтаксических деревьев были преобразованы в экземпляры, описанные в терминах выбранных нами свойств, они поступают на вход классификатора. На этапе обучения классификатор на основе тренировочных данных строит модель, которая затем используется на этапе применения классификатора для присвоения меток новым, тестовым экземплярам. Существует множество методов построения классификационных моделей, большая часть из которых более или менее успешно была применена для автоматической классификации актантов. Так, в работе [Johansson, Nugues, 2007] в качестве классификатора используется метод опорных веторов (SVM), а [Ngai и др., 2004] проводит сравнение систем на основе бустинга, метода опорных веторов, нейронных сетей и правил. Семантические метки актантов зависят друг от друга, т.е. с теоретической точки зрения правильным решением является использовать классификатор, который присваивает роль не отдельно каждому актанту, а всему набору актантов данного предиката одновременно.

Существуют техники машинного обучения, которые позволяют учитывать взаимозависимость семантических ролей [Das и др., 2010], однако эта взаимозависимость может быть включена в систему и в составе компонента постобработки, именно такой принцип был выбран в нашей системе.

В описываемой системе классификация осуществлялась с помощью метода опорных векторов (Support vector machines, SVM). Этот метод успешно применяется для решения многих задач в области автоматической обработки языка и имеет несколько преимуществ, которые делают его особенно интересным для нашей задачи. Метод имеет и недостатки, однако прежде чем мы обратимся к этой теме, кажется разумным в общих чертах описать принципы работы этого семейства классификаторов. Поскольку данное исследование не имеет своей целью детальный анализ работы алгоритмов машинного обучения применительно к задаче автоматической классификации актантов, мы ограничимся лишь полуформальным описанием принципов работы выбранного алгоритма. Метод опорных векторов хорошо исследован и описан в литературе, более формальное описание используемых алгоритмов содержится, например, в [Cortes, Vapnik, 1995].

Ранее в разделе, посвящённойм спецификации поставленной задачи и методов ее решения, был приведен пример более общей задачи классификации, в котором точки в признаковом пространстве координат разделялись на два класса и было необходимо построить решающую функцию, которая позволила бы для новых экземпляров определить, к какому классу они относятся. Мы вновь используем этот пример для того, чтобы продемонстрировать проблему, которую решает выбранный нами метод опорных векторов, и принцип его работы.

Итак, пусть наши экземпляры – это набор точек в двухмерном пространстве, и задача классификатора состоит в том, чтобы построить прямую,

по одну сторону от которой находятся точки одного класса, а по другую – точки другого класса.

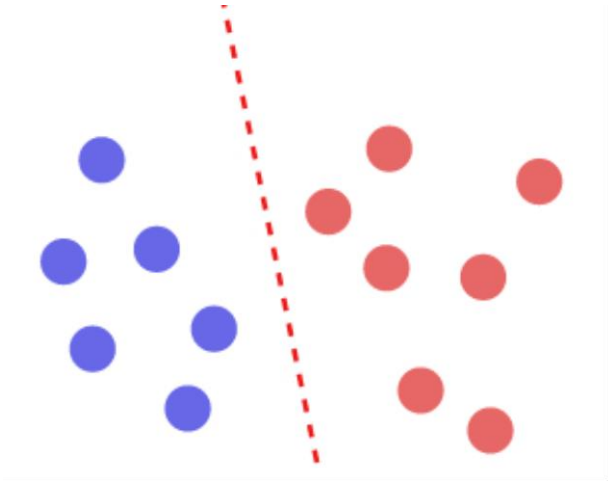

Рисунок 8: Решающая функция в двумерном пространстве

Можно заметить, что прямая, приведённая на Рис. 8, далеко не единственная из возможных, и исходя из тренировочных данных можно построить бесконечное количество прямых, которые удовлетворяют приведённому выше условию.

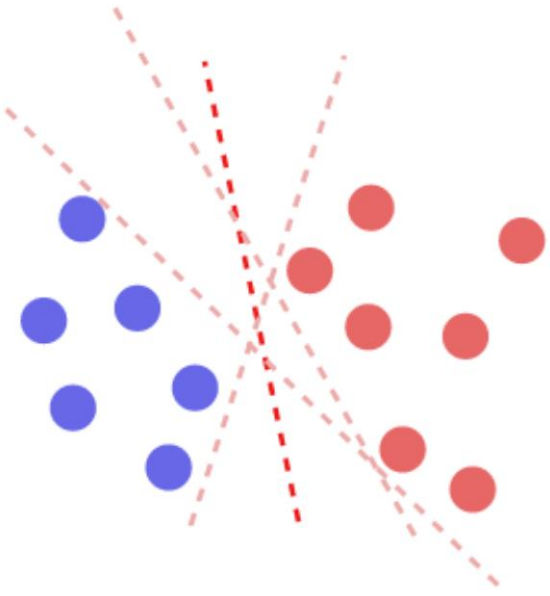

Рисунок 9: Альтернативные варианты построения решающей функции

Какая из прямых наилучшим образом разделяет наше признаковое пространство? С точки зрения способностей классификатора к генерализации, предпочтительной кажется прямая, которая разделяла бы пространство максимально надёжно, т.е. максимально удалённая от обеих множеств. Формализация этого требования и приводит нас к методу опорных векторов, суть которого заключается в следующем: требуется на основе тренировочных данных построить оптимальную разделяющую гиперплоскость (в нашем примере – прямую), которая максимально удалена от экземпляров, наиболее близких к границе класса. Проиллюстрируем эту идею примером:

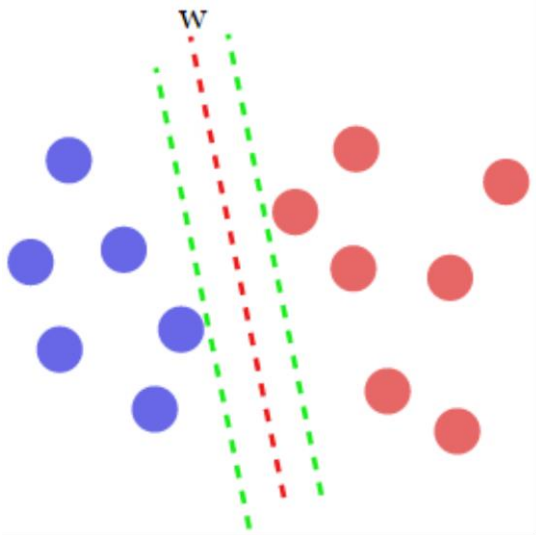

Рисунок 10: Решающая функция с максимальной границей

В данном примере прямая w разделяет признаковое пространство на два подпространства, причем построена она таким образом, что расстояние от прямой до ближайших к ней точек каждого класса максимизировано.

Опишем вышесказанное более формально. Пусть нам дан набор экземпляров, принадлежащих к одному из двух классов. Любая гиперплоскость в признаковом пространстве из d признаков может быть описана формулой вида $w \cdot x + b = 0$, где w – вектор, перпендикулярный гиперплоскости, а b –

константа, задающая расстояние от гиперплоскости до центра координат. Гиперплоскость, которая разделяет экземпляры на два класса, может быть использована в целевой функции $f(x) = sign(w \cdot x + b)$: если функция принимает положительное значение, экземпляр x получает метку первого класса, если значение отрицательное – приписывается метка второго класса. Поскольку, как мы уже продемонстрировали, таких гиперплоскостей можно построить бесконечно много, мы накладываем дополнительное ограничение: наша гиперплоскость должна быть максимально удалена от ближайших к ней экземпляров обоих классов. Расстояние от гиперплоскости до экземпляра x_i выражается формулой $d\big((w,b),x_i\big) = \frac{y(x_i \cdot w + b)}{\|w\|} \geq \frac{1}{\|w\|}$ и наша задача – максимизировать это расстояние. Как мы можем видеть, в данном случае достаточно будет максимизировать величину $1/\|w\|$, что эквивалентно минимизации $\|w\|$. При этом необходимо, чтобы все экземпляры из обучающей выборки были классифицированы корректно. Формально это ограничение выглядит как $y_i(x_i \cdot w + b) \leq 1\ (\forall\ i)$. Если объект принадлежит к первому классу ($y_i > 0$), то данное неравенство будет соблюдаться только в случае когда $y_i > 0$ и $x_i \cdot w + b > 0$, аналогичное верно для второго класса.

Эта задача может быть переформулирована как задача оптимизации и решена с помощью методов квадратического программирования. Традиционно (для удобства вычислений) минимизации подвергается не непосредственно $\|w\|$, а $\frac{1}{2} * \|w\|^2$. В этом случае задача оптимизации принимает следующий вид:

Минимизировать $\frac{1}{2}\sum_i \|w\|^2$

При условии $y_i(w \cdot x_i + b) - 1 \geq 0$

Для всех экземпляров $i = 1..N$

Обратим внимание, что целевая функция выпуклая, а это значит, что найденное решение (в данном случае, минимум), будет глобально оптимальным, т.е. решающая функция, созданная методом опорных векторов, будет единственной.

Метод опорных векторов в озвученной выше формулировке имеет ряд свойств, которые делают его интересным в рамках стоящей перед нами задачи. Обратим внимание, что положение разделяющей гиперплоскости зависит только от опорных векторов. Это позволяет значительно уменьшить объём данных, используемых для построения решающей функции, и выгодно отличает метод опорных векторов от, например, линейной регрессии и байесовских методов классификации, особенно в контексте, когда признаковое пространство имеет большую размерность.

Однако в данной формулировке метод опорных векторов восприимчив к шуму в исходных данных. Представим себе, что наши данные содержат несколько "ошибочных" экземпляров, расположенных в непосредственной близости от решающей гиперплоскости:

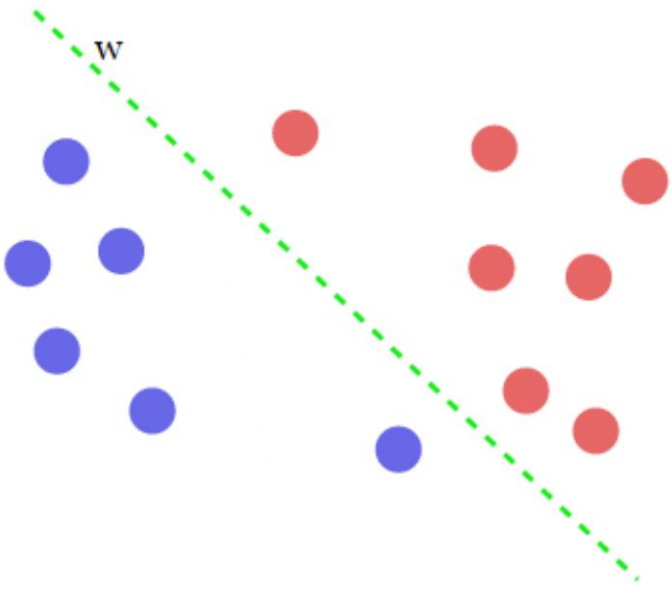

Рисунок 11: Проблемы решающих функций с жёсткой границей

Использование метода опорных векторов с жёсткой границей, который мы только что рассмотрели, может привести к построению неоптимальной гиперплоскости, т.к. учитывает только опорные, т.е. "проблемные" точки.

Решением в данном случае является модификация метода опорных векторов, которая разрешает неправильную классификацию некоторых экземпляров в тренировочных данных – классификаторы с мягкой границей. Этот подход позволяет повысить способность алгоритма к построению классификаторов, менее восприимчивых к ошибкам во входных данных. В задачу вводится дополнительный параметр C, который определяет стоимость ошибочной классификации на тренировочных данных.

Теперь задача оптимизации выглядит следующим образом:

Минимизировать $\frac{1}{2}\sum_i \|w\|^2 + C\sum_i \varepsilon_i$

При условии $y_i(w \cdot x_i + b) - 1 + \varepsilon_i \geq 0$

Здесь ε_i – вспомогательная переменная, которая равна расстоянию до гиперплоскости в случае, если экземпляр классифицирован неправильно, и 0 в случае правильной классификации. При высоких значениях параметра C поведение такого алгоритма приближается к стандартному методу опорных векторов, т.к. повышается стоимость ошибочной классификации. В целом, классификаторы на основе метода опорных векторов с мягкой границей менее восприимчивы к ошибкам в тренировочных данных и лучше подходят для решения нашей задачи с учётом того, что наши входные данные, несмотря на фильтрацию, содержат неточности.

Наш краткий обзор будет неполным без упоминания функций ядра, которые позволяют методу опорных векторов работать с линейно неразделимыми данными. Представим себе следующий набор данных:

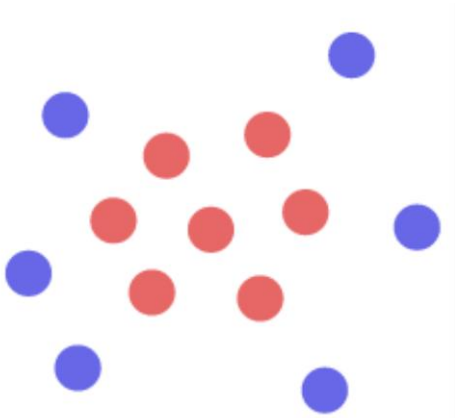

Рисунок 12: Линейно неразделимые классы

В данном случае найти прямую, которая разделяла бы точки, более не представляется возможным. Решением в данной ситуации является использование функции, которая переводит признаковое пространство в другое пространство, в котором точки оказываются разделимы с помощью гиперплоскости. Так, в приведённом выше примере мы можем использовать радиальную функцию преобразования, в результате применения которой точки оказываются разделимы:

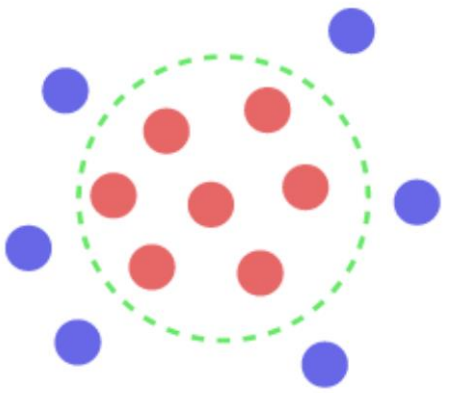

Рисунок 13: Разделение классов с помощью радиального ядра

В представленном исследовании мы не использовали преобразований пространства, т.к. их семантика в нашем случае непрозрачна. В силу того, что метод опорных векторов может работать только с числовыми свойствами, а наши атрибуты имеют номинальные значения, мы вынуждены перед классификацией произвести **бинаризацию признаков**. Суть этой процедуры может быть прояснена с помощью следующего примера. Рассмотрим признак "Лемма". Этот признак для каждого экземпляра принимает одно значение, например, *“машина”*, *“дом”*, *“красный”* и т.д. В процессе бинаризации мы заменяем признак "Лемма" на n бинарных признаков "*Лемма=машина*", "*Лемма=дом*" и т.д., где n – размер словаря значений выбранного признака. В результате каждый экземпляр оказывается описан в терминах набора бинарных признаков и может быть проинтерпретирован классификатором и использован для построения оптимальной гиперплоскости. Это признаковое пространство очень трудно визуализировать и интерпретировать, и мы не имеем возможности определить, требуется ли какое-либо преобразование в нашем случае, и если да, то какое. В связи с этим мы остановили выбор на линейном методе опорных векторов: в этом методе преобразований признакового пространства не производится, кроме того, он отличается высокой вычислительной эффективностью.

Метод опорных векторов в классической формулировке подходит только для решения задач бинарной классификации, однако наша задача в общем случае состоит в приписании одного класса-семантической роли из набора неопределённого размера (различные конструкции имеют разное число ролей). Для того, чтобы мы могли использовать SVM, мы должны интерпретировать задачу присвоения ролей из ролевого набора для выбранной конструкции в терминах бинарной классификации. Существует несколько способов сделать это, и мы рассмотрим два наиболее популярных из них.

Первый способ носит название "*один против всех*", и состоит в следующем: для каждого класса в тренировочном наборе данных мы строим классификатор, принимающий решение о том, принадлежит ли экземпляр к этому классу или нет. Затем каждый из K натренированных классификаторов применяется к новому экземпляру, и выбирается класс, получивший наибольший вес.

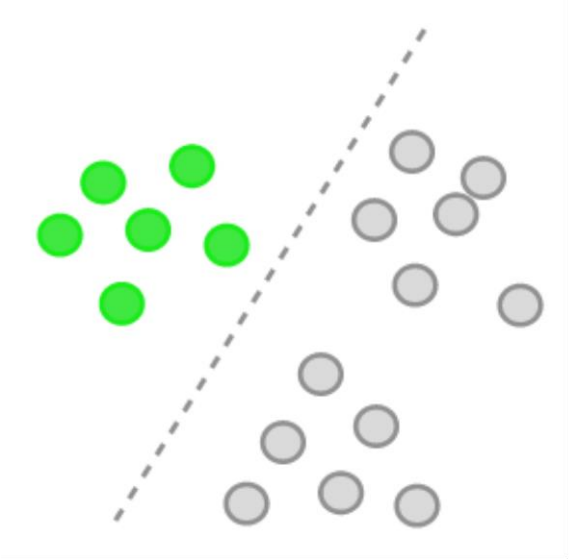

Рисунок 14: Классификация методом "один против всех"

Альтернативный способ приведения задачи мультиклассовой классификации к задаче бинарной классификации – "каждый против каждого". В этом случае для каждой пары классов строится отдельный классификатор (в сумме $K(K-1)/2$ классификаторов для K классов), и результаты работы этих классификаторов используются для **ранжирования** меток классов для каждого входного экземпляра. Несмотря на то, что в данном случае требуется построить больше классификаторов, для обучения каждого из них используется меньше данных, а решающие функции зачастую оказываются более надёжными. Кроме того, данный метод позволяет получить информацию о ранжировании.

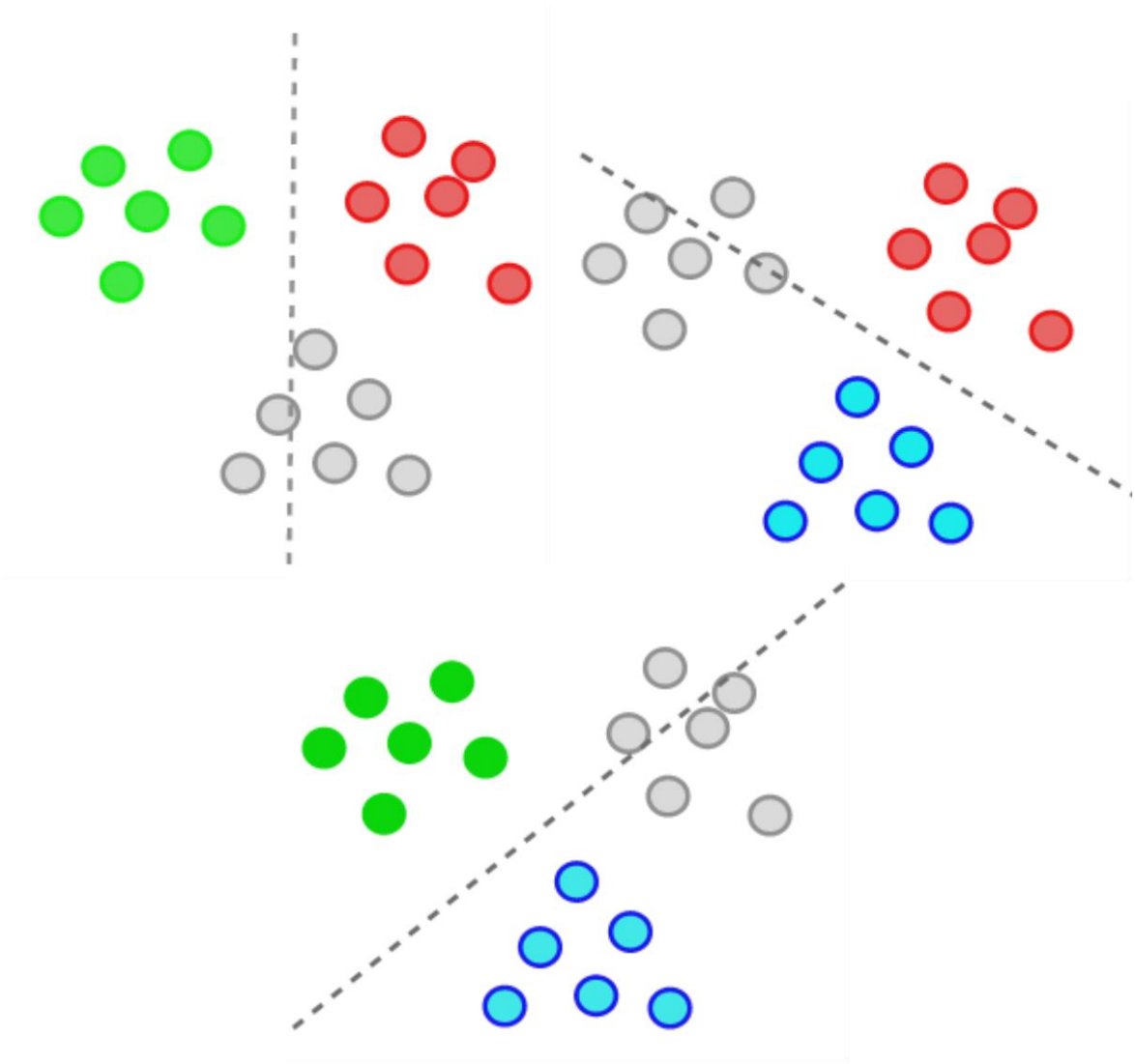

Рисунок 15: Классификация методом "каждый против каждого"

На практике выбор того или иного метода зависит от параметров задачи и от технических возможностей. В нашем исследовании мы остановились на классификации "один против всех", но это связано в первую очередь с особенностями имплементации нашей системы, а не с характеристиками задачи, в контексте которой, впрочем, мы не видим очевидных преимуществ использования попарной классификации над выбранным нами методом.

В заключение хочется отметить, что существует множество алгоритмов классификации, которые могут быть применены к задаче автоматической разметки актантов. В рамках настоящего исследования мы стремились сделать

систему максимально простой и при этом содержащей все необходимые компоненты и не проводили детального сравнения результатов работы различных классификаторов на наших данных. Одним из конкурирующих подходов при выборе алгоритма классификации в нашем случае были **деревья принятия решений**, преимущество которых состоит в том, что полученные модели могут быть легко интерпретированы человеком. На практике оказалось, что наши исходные данные содержат неточности, и деревья принятия решений работают на них неэффективно. Подбор и оптимизация классификатора безусловно имеют большое значение для задач, связанных с машинным обучением, однако поскольку основания выбора классификатора лишь косвенно связаны с лингвистическими задачами исследования, эта проблематика осталась за рамками настоящей работы. Существенно более значимым с точки зрения лингвистических оснований решения поставленной задачи является проблема выбора тех свойств, которые используются для обучения системы. Именно этому вопросу и будет посвящен следующий раздел.

II.3.4 Свойства для обучения

В качестве свойств, используемых для представления объекта, мы используем свойства, традиционно применяемые в системах автоматической классификации актантов на основе машинного обучения. Каждый узел дерева зависимостей мы описываем в терминах восьми свойств, объединённых в две группы: синтаксические и семантические свойства. Остановимся на них подробнее.

Синтаксические свойства

Задача синтаксических свойств – предоставить классификатору информацию о месте узла в синтаксическом дереве зависимостей и об индивидуальных синтаксических характеристиках узла, которые могут быть полезны для определения его семантической роли.

Свойство "путь" (Path)

Данное свойство мы определяем как путь от целевого предиката к рассматриваемому узлу в дереве зависимостей. Путь задаётся в терминах синтаксических отношений, а также направления этих отношений. Например, рассмотрим предложение "*Петя работает на заводе".*

Здесь "работать" является целевым предикатом конструкции. Путь от предиката к узлу "завод" будет состоять из двух связей с метками "2-компл" (второе комплетивное отношение) и "предл" (зависимость от предлога) или, более кратко, [2-компл, предл]. Вся структура представлена в виде дерева зависимостей на Рис. 16:

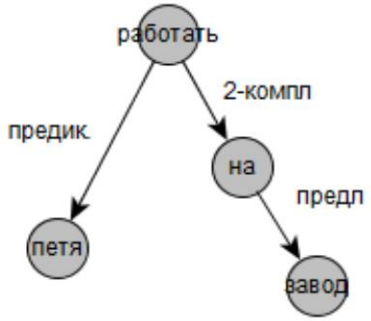

Рисунок 16: Дерево зависимостей

Поскольку семантические актанты не всегда соответствуют синтаксическим, путь от предиката до его семантического актанта может иметь существенную длину и выходить за пределы клаузы. Тем не менее, следует

отметить, что поскольку граф зависимостей по определению является деревом, оказывается возможным проложить путь между двумя любыми узлами дерева, в крайнем случае, для этого используется абстрактный корневой узел ROOT.

Путь – одно из самых важных и проблемных свойств в автоматической разметке актантов, и мы подробно остановимся на его особенностях ниже.

Свойство "падеж" (Case / FinnCase)

Русский язык использует падеж для маркирования синтаксических связей между предикатом и его синтаксическими аргументами. Несмотря на то, что семантические аргументы не всегда соответствуют синтаксическим, как правило, свойство "Падеж" предоставляет классификатору важную дополнительную информацию о маркировании выбранного узла. Например,

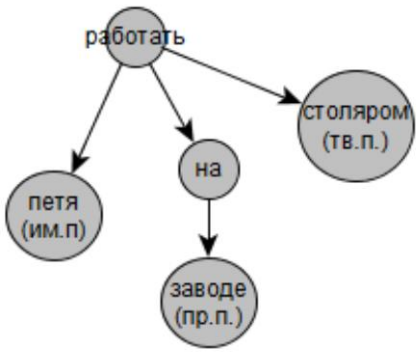

Рисунок 17: Дерево зависимостей без именованных отношений

В случае с падежным маркированием глагольных аргументов в русском языке, часто оказывается, что значение имеет не столько сам падеж слова, сколько комбинация предлога и падежа, которую мы далее будем именовать "финским падежом" и которой посвящён отдельный раздел.

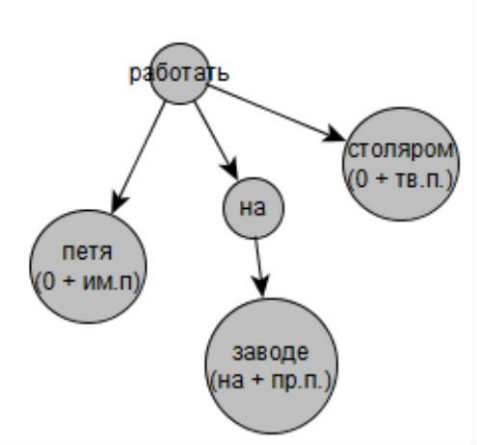

Рисунок 18: Дерево зависимостей с указанием "финского падежа"

Следует дополнительно отметить, что значимость свойств "падеж" и "финский падеж" возрастает в случае, если деревья зависимостей для обучающих и тестовых данных получены автоматически. Это происходит потому, что задача автоматического синтаксического анализа оказывается в целом сложнее, чем задача морфологического анализа, а качество работы синтаксических парсеров в целом ниже, чем качество морфологических анализаторов. В этой связи свойства, основанные на падеже, оказываются более надёжными и дают классификатору возможность принять правильное решение даже в тех случаях, когда синтаксическое дерево для тренировочного или тестового предложения было построено неверно.

Семантические свойства

Данная группа свойств отражает семантические характеристики лексемы, представленной в узле. Одно из свойств семантических ролей состоит в том, что они заполняются близкими по своим семантическим свойствам актантами. Действительно, можно представить себе случаи, когда даже одной только лексической информации о классах слов было бы достаточно для правильно интерпретации предложения в терминах семантических ролей. Рассмотрим следующий пример:

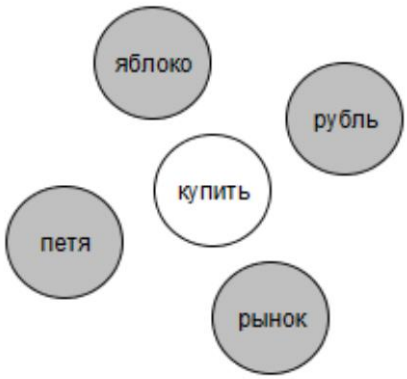

Рисунок 19: Представление на основе лексем

При условии, что нам известны семантические ограничения на заполнение ролей (в эксплицитной форме или в форме модели), даже при отсутствии информации о порядке слов и падежном маркировании. мы можем однозначно установить, что “Петя” является покупателем, “яблоко” – товаром, “рынок” – местом, а “рубль” – ценой.

На практике такие примеры встречаются нечасто. Кроме того, семантическая модель, которая позволила бы производить такой анализ в общем случае, отличается от традиционных моделей, построенных по принципу тезауруса, т.к. опирается на более тонкие, часто предикатно-специфичные различия между лексемами. В любом случае, семантические характеристики лексемы имеют большое значение для классификации актантов и традиционно входят в список наиболее важных для этой задачи свойств.

Свойство "Лемма"

Построение семантических моделей для описания ограничений на заполнение ролей – нетривиальная процедура. Однако приписание роли на основании буквального совпадения лексем в тренировочном и тестовом

предложении – достаточно надёжная и точная эвристика. Поскольку наша система не использует механизмов автоматического снятия лексической неоднозначности, и поскольку данный тип неоднозначности не снят в исходных данных, мы используем лемму слова, т.е. его начальную форму, например, “*покупателям*” → “*покупатель*”, “*синего*” → “*синий*” и т.д.

Свойство "Кластер"

Описание семантики актантов в терминах лемм позволяет добиться высокой точности при совпадении леммы, однако ведёт к потерям в полноте, т.к. буквальное совпадение лемм в общем случае маловероятно. Для того, чтобы смягчить этот эффект, как правило, используется внешний ресурс, содержащий, в том числе, обобщенную информацию как о леммах из тренировочного набора, так и о леммах из тестовых предложений: это может быть метка значения (как, например, в тезаурусах) или метка кластера. Таким образом, все леммы объединяются в определенные группы по семантической близости значений. В нашем исследовании мы использовали разбиение лексики на кластеры, автоматически полученное на большом объёме текстовых данных с помощью метода дистрибутивного семантического анализа [Harris, 1954; Mikolov и др., 2013]. Таким образом, вместо того, чтобы сравнивать конкретные леммы их обучающего и тестового корпуса, мы можем сравнить их семантические кластеры, и при совпадении класса считать слова сходными. Выбранный метод и результаты кластеризации будут специально рассматриваться ниже, здесь же ограничимся примером использования кластеров при генерации свойств узла. На Рис. 20 показано то, как кластеры слов дают возможность расширения лексических свойств лемм, влияющих на идентификацию ролевой принадлежности узла.

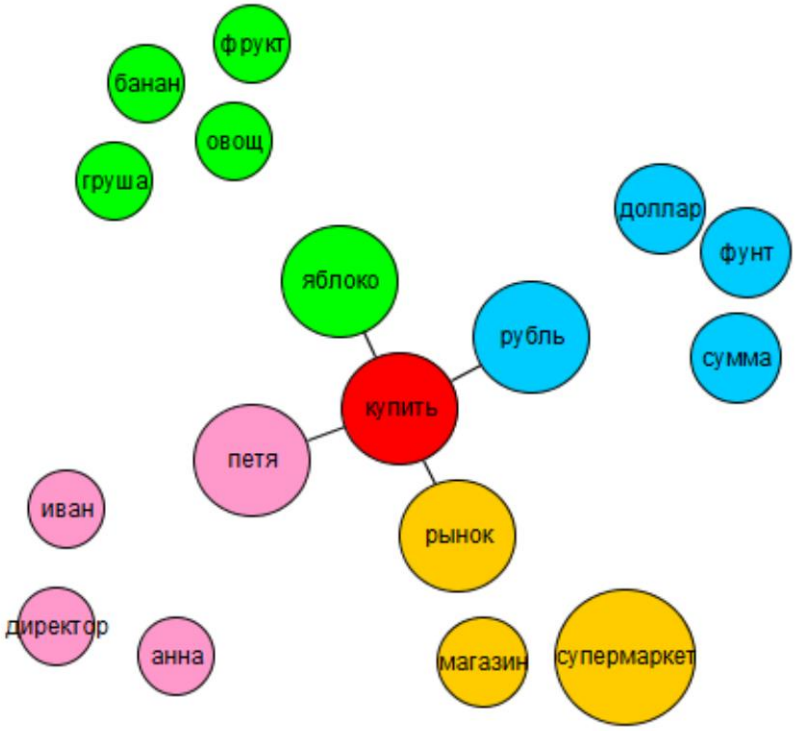

Рисунок 20: Представление на основе лексем с информацией о кластерах

Свойство "Часть речи"

Наконец, для описания узла мы используем его часть речи. Решение о том, чтобы считать это свойство семантическим, а не синтаксическим, может показаться спорным, однако с нашей точки зрения, в рамках задачи semantic role labeling для русского языка частеречная принадлежность слова несёт скорее семантическую, нежели синтаксическую нагрузку. В английском языке падеж в общем случае недоступен, и часть речи играет важную роль в том числе в качестве "подстраховки" в случае сбоев синтаксического парсера. В нашей системе эту роль играет падежное маркирование, часть речи же несёт более семантизированную нагрузку и служит, в частности, для разграничения имён существительных, наречий и сентенциальных актантов.

Итак, мы рассмотрели в общих чертах свойства, которые применяются в нашей системе для описания экземпляров. На основании этих свойств

классификатор принимает решение о том, какую семантическую роль (или её отсутствие) приписать каждому рассматриваемому узлу. Влияние каждого из этих свойств на результат классификации, было установлено в ходе экспериментов. Ниже мы хотели бы подробнее остановиться на нескольких свойствах, которые, на наш взгляд, представляют интерес вне зависимости от их вклада в качество работы системы в нашей имплементации.

II.3.5 Кластеризация лексики

Лексическая информация играет важную роль в автоматической классификации актантов и потому должна быть учтена в свойствах, используемых для описания экземпляров. Простейший способ учёта лексической информации – это непосредственно лемма слова, представленного целевым узлом. В случае совпадения леммы тренировочного и тестового экземпляра вероятность соответствия их ролей крайне высока. Рассмотрим следующий пример, где предложение из тренировочной выборки (слева) требуется сопоставить с предложением из тестовой выборки (справа):

Маша купила велосипед → Маша купила грузовик

В данном случае лексема "Маша" содержится в виде свойства как в тестовом экземпляре, так и в тренировочном, и на основании одного этого факта классификатор уже мог бы приписать тестовому узлу правильную роль. В то же время для второго узла, "грузовик", эта операция быть выполнена не может, т.к. его лемма не совпадает с леммой тестового экземпляра ("велосипед"). Поскольку объём корпусов, размеченных по семантическим ролям, как правило, очень ограничен, вероятность встретить новое слово в узле достаточно велика. Для того чтобы решить эту проблему, можно использовать внешний источник данных, который для каждой пары слов определяет, принадлежат ли они к одному семантическому классу. Имея такой ресурс,

созданный, как правило, с учётом значительно большего объёма данных, чем размеченный по семантическим ролям корпус, мы можем частично решить проблему низкого покрытия и делать успешные предсказания даже для узлов, лемма которых в тренировочных данных отсутствует.

Существует два основных типа ресурсов, которые могут быть использованы для решения этой задачи. Во-первых, можно использовать готовый внешний ресурс-тезаурус, созданный экспертами или полуавтоматически. В тезаурусе лексемы объединяются в группы, и для каждой пары лексем, в частности, можно установить, принадлежат ли они к одной и той же группе. В качестве иллюстрации приведём пример из тезауруса РуТез Lite [Loukachevitch, Dobrov, Chetviorkin, 2014]:

Рисунок 21: Запись тезауруса РуТез для концепта "Рептилия"

Такие ресурсы отличаются высоким качеством разбиения слов, однако могут страдать от недостаточной степени покрытия. Кроме того, ресурс может представлять классификацию слов в виде иерархии, в результате чего конкретный класс, к которому принадлежит слово, оказывается в действительности очень малочисленным. В этом случае требуется определить

некоторый уровень абстракции, на котором слова рассматриваются как принадлежащие к одному общему классу, и использовать классы на этом уровне абстракции в качестве свойств. Поскольку подобные ресурсы зачастую создаются с участием экспертов-аннотаторов, их объём, как правило, ограничен.

Альтернативное решение в данной ситуации – использовать результаты автоматической кластеризации лексики, полученной на большом корпусе данных. В общем случае задача кластеризации заключается в разбиении множества экземпляров на группы таким образом, что экземпляры внутри одной группы были максимально схожи между собой, при этом экземпляры из разных групп максимально различны. Задачу кластеризации иллюстрирует следующий пример, где выполняется объединение точек в группы на основании их расположения.

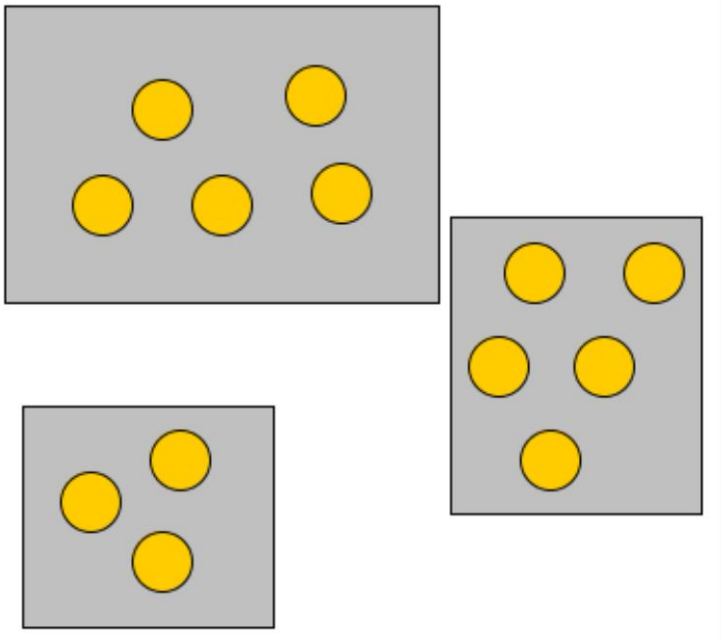

Рисунок 22: Задача кластеризации

В случае с кластеризацией лексики, точками-экземплярами являются отдельные лексемы или их значения. К задаче построения признакового

пространства существует несколько подходов, но все их объединяет предположение о том, что значение лексемы так или иначе выражается через множество её возможных контекстов. Наиболее простой вариант подобного пространства строится на основе частоты совместной встречаемости данной лексемы с другими лексемами. Однако было предложено множество альтернативных способов представления, которые позволяют автоматически группировать лексемы-признаки в соответствии с их распределением на исходном корпусе [Blei, Ng, Jordan, 2012; Gabrilovich, Markovitch, 2007; Mikolov и др., 2013]. После того как лексемы или значения представлены в виде экземпляров и описаны в терминах выбранного признакового пространства, мы можем выполнить кластеризацию этих точек и объединить их в группы на основании семантической близости. Полный обзор существующих методов кластеризации выходит за рамки задач этой работы, однако мы считаем уместным рассмотреть два наиболее распространённых подхода к кластеризации: плоскую кластеризацию, при которой лексемы распределяются по непересекающимся кластерам, и иерархическую кластеризацию, при которой кластеры организованы в иерархию.

Классическим представителем семейства алгоритмов плоской кластеризации является метод k-средних [MacQueen, 1967]. Принцип работы этого алгоритма состоит в том чтобы подобрать центры кластеров таким образом, чтобы минимизировать суммарное квадратичное отклонение точек кластеров от этих центров.

При инициализации алгоритмов выбирается k случайных точек признакового пространства – центров кластеров – и оставшиеся точки (исходные экземпляры) разделяются на кластеры в зависимости от того, к какой из инициирующих k точек они ближе расположены. После этого для каждого из k кластеров вычисляется центр масс на основе всех входящих в него точек, и этот центр масс объявляется новым "центром кластера". Эти действия

повторяются до того момента, когда очередное обновление центров масс не приводит к изменениям.

К недостаткам этого подхода относят зависимость результата от случайного выбора центров кластеров при инициализации и необходимость указать целевое число кластеров k (что в случае с кластеризацией лексики представляется затруднительным).

Классический представитель алгоритмов иерархической кластеризации – агломеративный алгоритм [Sibson, 1973]. Суть этого алгоритма состоит в том, что два наиболее схожих между собой кластера объединяются на каждом новом шаге в один. Изначально каждая точка признакового пространства является отдельным кластером. Для каждой пары точек мы вычисляем их сходство с помощью одной из стандартных мер близости векторов (например, косинусного расстояния) и объединяем наиболее близкие точки в кластер. Эта процедура повторяется, причём для кластеров при вычислении сходства используется центр кластера. Процедура останавливается, когда все точки оказываются объединены в один кластер. Результат агломеративной кластеризации – разбиение точек на группы, организованные в иерархию. Сложность с использованием данного подхода в задачах, подобных нашей, состоит в том, что одной метки кластера для слова недостаточно и необходимо каким-то образом передавать в классификатор информацию об иерархических отношениях внутри множества кластеров. В то же время, как и в случае с тезаурусами, при нахождении оптимального порога отсечения и группировки оказывается возможным получить разбиение лексики на осмысленные классы, которые могут использоваться при автоматической разметке актантов.

В связи с тем, что учёт иерархических отношений требует дополнительного моделирования, в данном исследовании мы остановили выбор на плоской кластеризации, которая приписывает каждой лемме только один семантический класс. Поскольку изначальное число кластеров для

плоской кластеризации определить трудно, мы используем непараметрический графовый алгоритм кластеризации Chinese Whispers, предложенный в [Biemann, 2006a]. Нам не известно о случаях применения этого алгоритма для решения указанной задачи в русском языке, однако мы считаем полученные нами результаты обнадёживающими, что поддерживается и высоким качеством результатов, полученных для аналогичной задачи на английском материале. Поскольку выбранный нами алгоритм появился сравнительно недавно и используется не очень широко, представляется разумным коротко остановиться на общих принципах его работы. Это кажется тем более уместным, что выбранный нами алгоритм достаточно прост и интуитивно понятен.

Chinese Whispers является алгоритмом кластеризации графов. Суть его демонстрирует следующий пример. Допустим, что нам дан граф, состоящий из узлов и ненаправленных взвешенных связей между ними. Задача состоит в том, чтобы сгруппировать узлы на основании этих связей. Для простоты предположим, что веса всех связей равны единице.

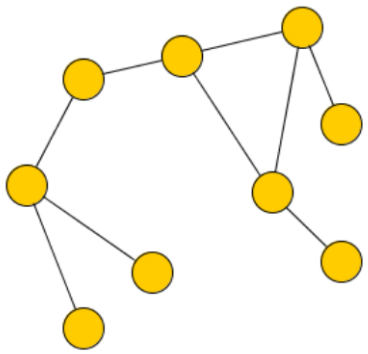

Рисунок 23: Пример графа

На этапе инициализации каждый из узлов графа получает уникальную метку кластера.

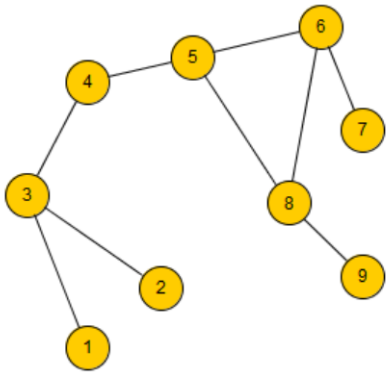

Рисунок 24: Граф после инициализации

Затем в ходе каждой итерации каждый узел *голосует* за свою метку кластера с силой, равной весу связи. Таким образом для каждого узла метка его кластера определяется суммой голосов за каждую из меток соседствующих узлов. Порядок обхода узлов определяется случайно в начале каждой итерации, в случае неоднозначностей решение принимается случайно. Представим, что в определенный момент времени граф оказался в следующей конфигурации, и в настоящий момент выполняется голосование за метку для узла *5*.

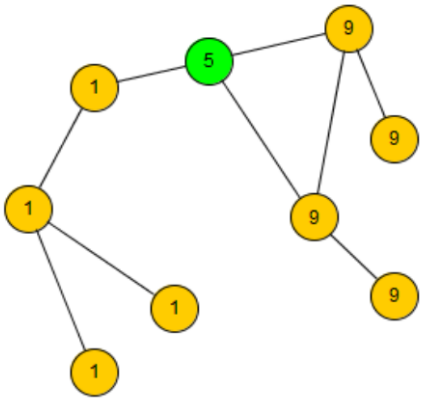

Рисунок 25: Шаг кластеризации: голосование

В результате голосования, метка *9* получает два голоса, т.к. эту метку имеет два узла, связанных с узлом *5*. За метку *1* голосует только один узел, таким образом, метка рассматриваемого узла будет изменена с *5* на *9*.

Более формально алгоритм может быть представлен следующим образом:

Инициализация: для каждого узла $v_i \in V: class(v) = i$
Пока есть изменения: для каждого $v_i \in V$ в случайном порядке $class(v_i)$ устанавливается как класс с наиболее высоким рангом среди соседей узла v_i в графе. При неоднозначности класс выбирается случайно.

В результате, по истечении определённого числа итераций, достигается оптимальное жёсткое разделение графа на кластеры. Количество итераций определяется эмпирически, но авторы метода заявляют, что для достижения оптимального разбиения графа требуется всего несколько итераций, и после 10 итераций разбиение меняется незначительно или не меняется вовсе [Biemann, 2006a]. Авторы алгоритма экспериментально установили, что оптимальное число итераций равно 10, это значение используется в имплементации алгоритма по умолчанию, и мы также будем придерживаться этой величины.

Для того чтобы конвертировать наше исходное множество точек-лексем в графовое представление мы используем метрику семантической близости на векторной модели RusVectōrēs [Kutuzov, Andreev, 2015], созданной с помощью инструмента word2vec [Mikolov и др., 2013] на основе большого корпуса русскоязычных новостных текстов. Данная модель описывает лексемы языка в терминах изменённого признакового пространства, построенного с использованием нейронных сетей. Данные, полученные с помощью word2vec, позволяют эффективно вычислять семантическую близость лексем, а также

выполнять дополнительные операции, например, "сложение" и "вычитание" смыслов (анализ этого явления см. в [Levy и др., 2015]).

Используя представление RusVectōrēs, мы получаем для каждого слова, содержащегося в модели, 10 наиболее похожих на него слов с их весами, которые в выбранной нами имплементации вычисляются как косинусная мера сходства. Затем все слова модели помещаются в граф в качестве узлов, и в случае, если одно из слов вошло в список 10 наиболее похожих для другого слова, эти слова связываются отношением с весом, равным степени их сходства. К полученному графу применяется алгоритм Chinese Whispers в стандартной конфигурации, и в результате работы этого алгоритма каждая лексема получает метку кластера, которая может быть использована в качестве свойства в нашем классификаторе. Следующие примеры иллюстрируют качество работы выбранного нами метода.

Кластер 85	**Кластер 418**	**Кластер 81**
спица	монарх	раменки
колесико	кронпринц	гольяново
ограничитель	елизавета	зябликово
светотехника	уильям	вешняк
сидушка	маргрета	ростокино
поворотник	герцог	пресня
царапина	королева	бескудниково
подкрылок	высочество	силино
ездок	чарльз	капотня
фонарь	престол	лианозово
стойка	наследник	дегунино
индикация	трон	замоскворечье
педаль	принц	марьина
дверка	бурбон	хамовник
воздухообмен	престолонаследник	шумак
кресло	величество	новогиреево
джерардо	корона	митино
боковой	монархия	челобитьево
переносок	король	лихобор
...	...	...

Рисунок 26: Пример полученных кластеров

Как мы можем видеть, кластеры имеют достаточно однородный состав. Исходные данные RusVectōrēs содержат языковой материал из других языков, в частности, из белорусского. Однако дополнительный эффект кластеризации состоит в том, что слова из других языков контекстно схожи и группируются в отдельные кластеры, и, таким образом, оказываются изолированы от принятия решений по присвоению семантических ролей. В то же время отметим, что кластеризация английских лексем выполняется корректно, например, в один кластер объединяются лексемы, обозначающие компании.

В нашем исследовании мы используем две модификации описанного выше подхода. Исходные данные содержат представления как для имён существительных, так и для слов с другими частями речи. В некоторых случаях кластеризация, полученная с использованием всех лексем, не лишена смысла и может оказаться полезна, однако использование всех частей речи может приводить и к нежелательным эффектам из-за слияния кластеров с разными значениями за счёт соседства с узлом-глаголом или узлом-прилагательным.

В наших экспериментах мы используем два варианта кластеризации: в одном из них используются только имена существительные, другой же использует все лексемы, встречающиеся в исходных данных.

II.3.6 Детали реализации свойства "путь"

Синтаксическая структура предложения представляет собой формальное описание предложения, которое отражает синтаксические связи между его членами. Два наиболее популярных класса формализмов, используемых для этой задачи, это деревья непосредственных составляющих и деревья зависимостей.

Деревья непосредственных составляющих были предложены в рамках генеративного направления [Carnie, 2007]. В формализмах данного типа

предложение представляется в виде набора вложенных в друг друга структур-составляющих, каждая из которых обладает относительной внутренней автономией. Структура предложения представляется в виде дерева составляющих, состоящего из нетерминальных (фразовых) и терминальных узлов. Нетерминальные узлы объединяют дочерние компоненты в т.н. группу, например, именную или глагольную. Терминальные узлы содержат непосредственно слова предложения и не имеют потомков. Следующий пример иллюстрирует формализм дерева составляющих для английского языка.

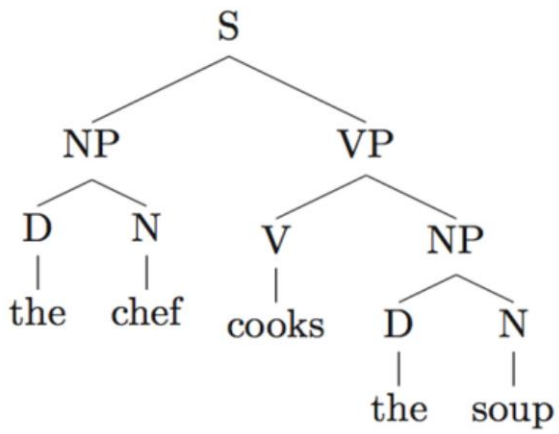

Рисунок 27: Дерево НС для английского языка

Формализм непосредственных составляющих имеет определённые преимущества, однако разрабатывался в первую очередь для английского языка и не всегда позволяет компактно описать синтаксическую структуру в других языках. В частности, трудности для формализма НС представляют языки со свободным порядком слов (что ведёт к разрыву составляющих) и с зачастую сопутствующим ему падежным маркированием синтаксических отношений (в результате чего возникает необходимость в использовании промежуточных узлов).

В качестве альтернативы для языков со свободным порядком слов и падежным маркированием используется синтаксис деревьев зависимостей [Mel’čuk, 1988]. Формализм деревьев зависимостей также предполагает построение графа синтаксических отношений между словами предложения, однако в отличие от дерева НС не является иерархическим. В основе формализмов зависимостей лежит граф, к которому применяются следующие требования. Граф содержит направленные отношения между словами предложения, от главного к зависимому. У каждого слова должен быть только один и только один "родитель", и граф не должен содержать циклов. Для обозначения корня синтаксического дерева вводится специальный служебный элемент, который является родителем главного слова в предложении. Синтаксические отношения могут быть именованными, но это не является обязательным требованием.

Данное представление является более компактным и гибким по сравнению с деревьями непосредственных составляющих, однако не позволяет напрямую обращаться к синтаксическим группам. В то же время для большинства задач автоматической обработки языка деревья зависимостей оказываются подходящим уровнем абстракции и помимо автоматической классификации актантов активно применяются в построении языковых моделей [Levy, Goldberg, 2014], расчёте семантической близости [Lin, 1998] и других задачах.

В завершение мы хотели бы отметить, что конвертация из деревьев зависимостей в деревья составляющих возможна почти всегда (при условии, что в дереве составляющих отмечаются главные слова группы), а выбор конкретного формализма зависит в первую очередь от доступности синтаксических анализаторов, лингвистических традиций для конкретного языка, а также предпочтений в рамках конкретной задачи.

Даже в рамках одного формализма может существовать множество вариаций, особенно если речь идёт об описании синтаксиса нескольких языков. Отличаться может набор синтаксических отношений (в случае деревьев зависимостей) и групп (для деревьев НС), а также частные правила установления отношений между словами. В качестве иллюстрации приведём 4 различных способа представления синтаксической структуры сочинительной группы, каждый из которых полностью отвечает требованиям формализма деревьев зависимостей:

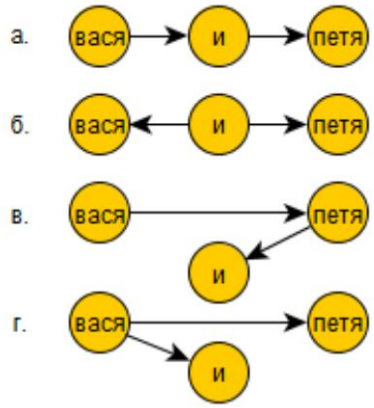

Рисунок 28: Различные варианты представления сочинительной группы

Другой пример – использование "компактных" зависимостей для предложных групп, которое мы можем встретить в синтаксическом анализаторе StanfordParser [Marneffe De, MacCartney, Manning, 2006] для английского и в парсере CognitiveDwarf [Мисюрев, Antonova, 2012] для русского языка:

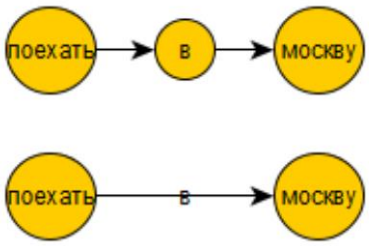

Рисунок 29: Компактные зависимости

Решение в подобных спорных случаях принимается исходя из конкретной синтаксической теории, на которую опирается исследователь. Следует отметить, что подобные описательные условности могут приводить к сложностям при использовании результатов работы автоматических синтаксических анализаторов для более высокоуровневых задач. При применении готового алгоритма, основанного на синтаксической структуре, необходимо убедиться, что синтаксическая модель, на основе которой разрабатывался алгоритм, и текущая синтаксическая модель совместимы.

В нашем исследовании мы опираемся на усовершенствованный формализм модели Смысл↔Текст [Мельчук, 1974], использованный в единственном на текущий момент синтаксически аннотированном корпусе для русского языка СинТагРус, разработанном ИППИ РАН (подробнее см. [Апресян, Богуславский, Иомдин, 2005]). Деревья зависимостей в рамках этого формализма представляют собой ациклические направленные графы с единственной абстрактной вершиной ROOT и именованными синтаксическими отношениями.

Анализ предложения в рамках выбранного нами формализма выглядит следующим образом:

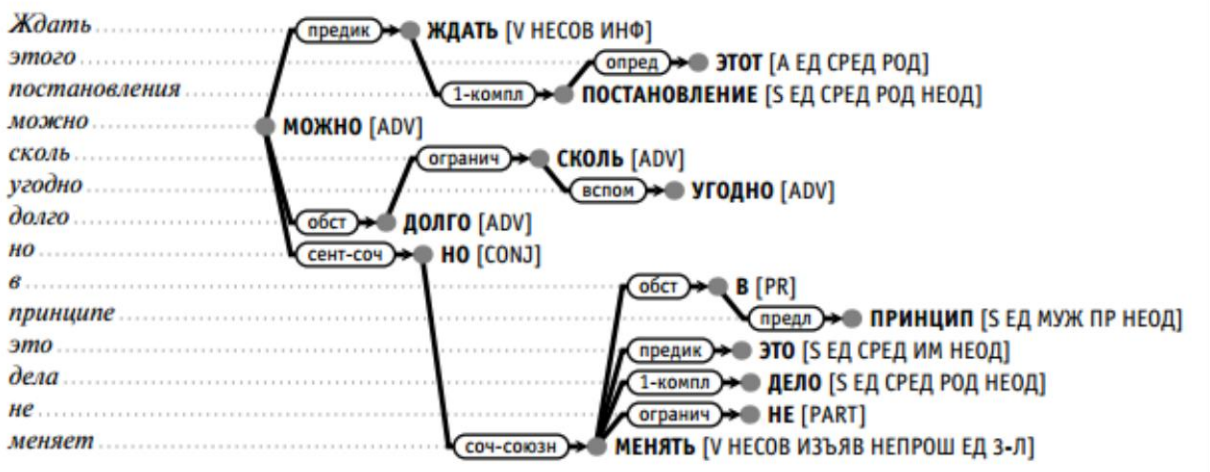

Рисунок 30: Анализ предложения в формализме СинТагРус

На примере этого предложения мы объясним и продемонстрируем принцип работы свойства "путь". В общем случае путь между двумя словами предложения определяется как последовательность синтаксических отношений в дереве зависимостей, которая маркирует кратчайший путь в графе зависимостей между этими словами. Для того, чтобы однозначно идентифицировать путь, мы дополняем имена синтаксических отношений информацией о направлении отношения. Благодаря тому, что граф ацикличен, имеет один корневой узел, и что при поиске пути мы можем перемещаться как в направлении отношения, так и в противоположном направлении, мы можем найти путь между двумя любыми словами предложения. Например, кратчайший путь от слова “принцип” к слову “долго” – предложное, обстоятельственное, сочинительно-союзное и сентенциально-сочинительные отношения против направления зависимости, и затем обстоятельственное отношение по направлению зависимости, или, кратко, [-предл, -обст, -соч-союзн, -сент-соч, обст], где знак минус обозначает обратное движение, т.е. от зависимого к главному.

Поскольку в контексте автоматической классификации актантов наибольший интерес представляет путь между целевым предикатом и потенциальным актантом, мы определяем свойство "путь" для каждого слова предложения как путь от предиката до этого слова. На этапе подготовки данных к классификации мы производим автоматический синтаксический анализ исходного предложения и вычисляем значение свойства "путь", которое затем используется при обучении и применении классификатора.

Так, для первых слов из указанного выше предложения с целевым предикатом “*ждать*” были бы извлечены следующие значения свойства "путь":

этого	1-компл, предл
постановления	1-компл
можно	-предик
долго	-предик, обст

Таблица 2: Значения свойства путь

В рамках нашей задачи рассматриваемое свойство имеет большую важность, т.к. имплицитно включает в себя информацию о субкатегориальной рамке предиката и о кореференции в случаях с удалёнными актантами. В случаях, если автоматический синтаксический анализ был произведён верно, совпадение пути, т.е. совпадение последовательности имён отношений с направлениями, является веским аргументом в пользу правильности гипотезы о соответствии семантических ролей. Однако учитывая, что синтаксический анализ в нашем случае производится автоматически как на этапе обучения классификатора, так и на этапе применения, при увеличении длины пути возрастает и вероятность, что данный путь был определён неправильно, что приводит к появлению большого числа уникальных "длинных" путей и общей разреженности данных.

Уникальные пути составляют около 50% всех обнаруженных путей на наших тренировочных данных, при этом польза этих уникальных путей сомнительна, так как обладая высокой специфичностью они могут быть ошибочными. В то же время вероятность, что путь содержит ошибку, растёт с увеличением длины пути. Исходя из этого, кажется разумным ограничить длину пути. Следующий график демонстрирует соотношения длин пути с их частотами в нашем корпусе:

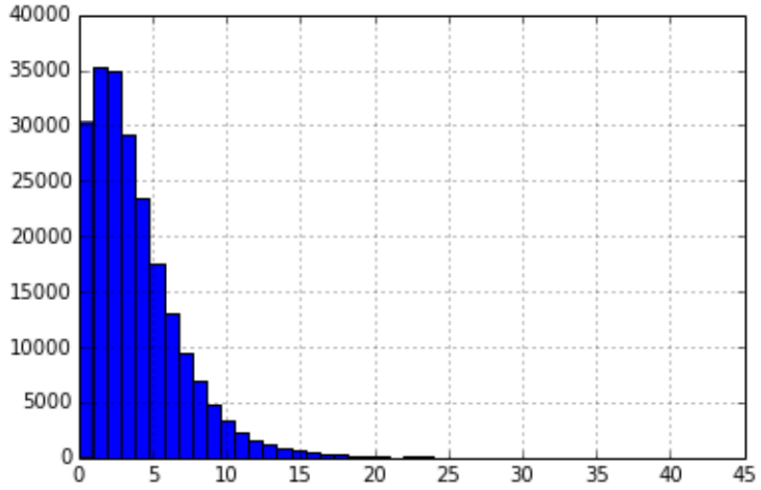

Рисунок 31: Гистограмма длины свойства "путь"

Пик распределения приходится на пути с частотой 1-4, затем частота падает. Средняя длина пути по нашей обучающей выборке составляет 3.5 отношения, поэтому было принято решение ограничить длину рассматриваемых путей 4 отношениями. Пути, длина которых превышает эту величину, могут быть искусственным образом сокращены до 4 шагов от предиката. В этом случае решение о присвоении роли может приниматься на основе других свойств выбранного экземпляра. Это свойство, которое мы в дальнейшем будем именовать path4, используется при классификации наравне со свойством path, которое представляет собой полный путь.

II.3.7 Свойство "финский падеж"

В языках со свободным порядком слов и падежным маркированием падеж часто используется для отражения синтаксических зависимостей между членами предложения. В некоторых случаях информации о падеже может быть достаточно для определения семантической роли того или иного актанта. Рассмотрим пример на Рисунок 32, в котором порядок слов и лексическая информация полностью удалены и единственная доступная информация – это

падеж и часть речи выбранного слова. Этому примеру может соответствовать предложение “*Директор купил синюю машину*”.

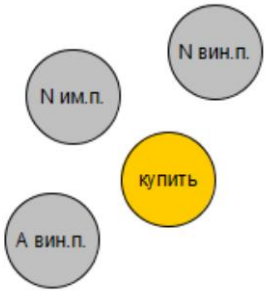

Рисунок 32: Представление предложения на основе информации о части речи и падеже

Человеку, знакомому с терминологией, не составит труда определить, что узел-существительное в именительном падеже – это Агенс или Покупатель, а существительное в винительном падеже – Товар.

В то же время система, которая опиралась бы только на информацию о падеже, столкнулась бы с трудностями по крайней мере в двух случаях. Во-первых, представим себе, что на вход системы поступило достаточно длинное предложение, включающее в себя несколько клауз (“*Получив премию, директор поехал в салон и купил себе синюю машину*”), см. Рис. 33:

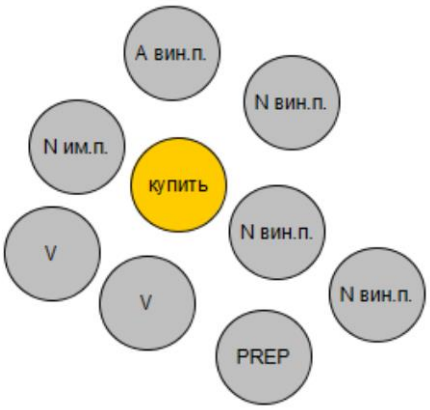

Рисунок 33: Недостаточность представления на основе части речи и падежа

В этом случае падежной информации для выполнения нашей задачи недостаточно. Кроме того, опираясь лишь на информацию о падеже и части речи, мы не можем однозначно определить, какое именно главное слово вызвало появление того или иного падежного маркера. Эта информация принадлежит к более высокому уровню анализа и в рамках нашей системы моделируется с помощью свойства "путь". Отметим, что в качестве альтернативы свойству "путь" мы могли бы использовать свойства, основанные на порядке слов, и отталкиваться от эвристического правила, согласно которому зависимое слово находится ближе к главному, что соответствует принципу проективности зависимых элементов.

Вторая сложность связана с тем, что в действительности информация о семантической роли слова зачастую кодируется не самим падежом, а комбинацией предлога и падежа. Можно было бы предположить, что для маркирования роли языку достаточно предлога, однако, например, в русском языке падеж существительного определяется предлогом неоднозначно, и один и тот же предлог с существительными в разных падежах может кодировать разные семантические роли.

Используя комбинацию предлога и падежа, мы можем добиться достаточно высокой описательной точности при описании ролевого маркирования слов, находящихся с предикатом в одной клаузе. Учитывая, что подобное маркирование является хорошим индикатором для нашей задачи, мы включили в набор свойств помимо непосредственно падежа как морфологической характеристики также "*финский падеж*". Этот термин мы используем неформально для обозначения выбранного нами типа маркирования. Своим происхождением "финский падеж" обязан финно-угорским языкам, в которых семантика падежных показателей включает в себя значения, которые в русском и многих других языках выражаются предлогами.

"Финский падеж" является морфо-синтаксической характеристикой и представляет собой конструкцию из предлога и падежного показателя зависимого имени. В большинстве случаев это свойство может быть выделено только для имён существительных, в случае же, когда падеж или предлог недоступны, свойство принимает специальное "пустое" значение (для классификатора это значение ничем принципиально не отличается от всех остальных).

В завершение отметим, что использование комбинации предлога и падежа не решает проблемы синтаксически удалённых актантов и может привести к ложным срабатываниям в тех случаях, когда целевое слово является зависимым некоторого другого слова, а не непосредственно предиката. В связи с этим мы можем ожидать, что свойство "финский падеж" будет наиболее эффективно срабатывать в связке со свойством "путь", контролирующим дистанцию, а также с лексическими свойствами (например, "кластер"), накладывающими ограничения на лексическое заполнение актантов выбранного предиката. В этом случае комбинация падежа и предлога играет важную уточняющую роль и, как мы увидим далее, оказывается более полезной, нежели изолированный падеж.

Итак, мы рассмотрели свойства, на основании которых система принимает решение о приписании семантических ролей. Мы используем богатый и лингвистически мотивированный набор свойств, которые мы извлекаем как в результате предобработки входных предложений (например, падеж или синтаксический путь), так и в результате обращения к внешним ресурсам (например, кластерная лексическая модель). Обобщая, использованные свойства можно разделить на две группы: синтаксические и семантические свойства. Вклад каждой из групп, а также отдельных свойств, в качество классификации оценивается по результатам применения алгоритма машинного обучения на основе различных комбинаций выбранных свойств.

Мы ещё вернемся к этому вопросу, когда будем подробно рассматривать результаты работы системы, а сейчас перейдём к описанию завершающего компонента нашей системы – модуля глобальной оптимизации.

II.4 Глобальная оптимизация разметки актантов

II.4.1 Задача глобальной оптимизации ролей

Итак, к настоящему моменту наша система представляет локальный классификатор, который на основе свойств приписывает каждому узлу синтаксического дерева зависимостей метку роли или специальную метку, обозначающую, что данный узел никакой ролевой нагрузки не несёт. Присвоение роли каждому узлу дерева происходит независимо, другими словами, при принятии решения о том, какую роль получает данный узел, классификатор не использует информацию о своих решениях для предыдущих узлов. Такая конфигурация делает возможной ситуацию, в которой несколько узлов в предложении получают одинаковую метку роли.

Существует несколько причин, по которым мы хотели бы подобной ситуации избежать. Во-первых, приписание одной и той же роли двум и более узлам противоречит одному из основных принципов теории семантических ролей, а именно, требованию, чтобы каждая роль заполнялась только одним актантом. Так, например, теория семантических ролей объясняет, почему невозможны конструкции типа **Петр ударил по столу молотком кулаком* и **Петр едет на юг на восток*, в то время как конструкция *Петр едет на машине на юг на конференцию* возможна (представленное здесь явление называется расщеплением актантов, см. [Апресян, 2006]).

В случае когда роль действительно заполняется несколькими реальными актантами, например, в сочинительных конструкциях, должен быть выбран только один из них, или же они должны быть объединены в группу и этот факт должен быть маркирован синтаксически (например, сочинением узлов).

Если наша система будет присваивать одну и ту же роль нескольким узлам, её результат будет затруднительно интерпретировать в рамках теории семантических ролей.

Кроме формальных причин избегать подобных ситуаций, есть причины и практического характера. Допущение о независимости ролей является очень серьёзным упрощением задачи. В действительности мы хотим, чтобы система не просто приписала каждому узлу уникальную семантическую роль, но и чтобы набор приписанных ролей для всего предложения был наилучшим среди возможных для данного классификатора. В качестве иллюстрации представим себе следующую ситуацию. Допустим, что мы выполняем анализ предложения “Петр купил яблоко за пять рублей” и наш ролевой инвентарь включает в себя роли X,Y и Z. В скобках укажем веса, которые классификатор приписывает каждому из классов в зависимости от свойств узла. Итоговое решение классификатора состоит в том, чтобы выбрать для каждого узла класс с наибольшим весом.

	X	Y	Z
Петр	**0.9**	0.1	0.1
яблоко	**0.8**	0.6	0.1
за	**0.8**	0.1	0.6
рубль	**0.7**	0.5	0.2

Таблица 3: Распределение весов при классификации

Как мы можем наблюдать, класс X имеет стабильно более высокий вес, чем остальные два класса, и поэтому будет приписан всем узлам. В результате мы не только получим несколько узлов с одной семантической ролью, что запрещено, но и потеряем информацию о двух других ролях. Мы можем потребовать, чтобы каждая семантическая роль встречалась только один раз, но в этом случае возникает вопрос, какой именно из узлов, или, в общем случае, какую комбинацию узлов следует выбрать. Логичным будет

предположить, что в случае, когда узлы нам уже даны, наилучшим решением будет выбрать комбинацию ролей, которая максимизировала бы "уверенность" классификатора в принятых решениях.

Решение данной задачи путём расчёта целевого значения для всех возможных вариантов приписания ролей крайне затратно с вычислительной точки зрения. Так, количество возможных разборов приведённого выше примера равно 64, или, в общем случае, n^m, где n – количество узлов в синтаксическом дереве предложении, а m – количество ролей, доступных для данной конструкции. Однако существуют более эффективные методы решения этой задачи, один из которых – метод целочисленного линейного программирования – был использован для разработки рассматриваемой системы. Описанию этого метода посвящён следующий раздел.

II.4.2 Линейное программирование: принцип работы

Целочисленное программирование (*Integer linear programming*, *ILP*) – частный случай линейного программирования. **Линейное программирование** – это парадигма, предназначенная для решения задач, которые могут быть представленны в следующей форме. Предположим, что нам дан набор переменных $x_1, x_2, \dots x_n \in x$. Необходимо максимизировать или минимизировать целевую линейную функцию $f(x) = x_1c_1 + x_2c_2 + \cdots + x_nc_n$ с учетом набора линейных ограничений вида $a_{11}x_1 + a_{12}x_2 \geq b_1$, $a_{21}x_1 + a_{22}x_2 = b_2$ и т.д., при этом все переменные принимают только положительные значения:

Максимизировать: $x_1c_1 + x_2c_2$

С учетом ограничений:

$a_{11}x_1 + a_{12}x_2 \geq b_1$

$a_{21}x_1 + a_{22}x_2 \geq b_2$

$a_{31}x_1 + a_{32}x_2 \geq b_3$

$x_1 \geq 0, x_2 \geq 0$

Задача может быть переформулирована в векторной форме:

Максимизировать: $C^T x$

С учетом ограничений:

$Ax \leq B; x \geq 0$

В качестве классической иллюстрации задачи, решаемой с помощью линейного программирования, приводят задачу об оптимизации прибыли при условии ограниченных ресурсов. Предположим, что фермер хочет засеять площадь в 8 га пшеницей и кукурузой. При этом с каждого гектара, засеянного пшеницей, он получает 5000 рублей прибыли, а за каждый гектар кукурузы — 3000 рублей. Количество пестицидов, которые он может использовать, ограничено 10 литрами. При этом на 1 гектар пшеницы требуется 2 литра пестицида, а на 1 гектар кукурузы — только один литр. Необходимо определить, какую площадь требуется засеять картофелем, а какую – пшеном, чтобы максимизировать прибыль. Сформулируем задачу в терминах ЛП:

x гектаров пшеницы, y гектаров кукурузы

Максимизировать: $5000x + 3000y$

С учетом ограничений:

Ограничение на пестициды $2x + 1y \leq 10$

Ограничение на площадь $x + y \leq 8$

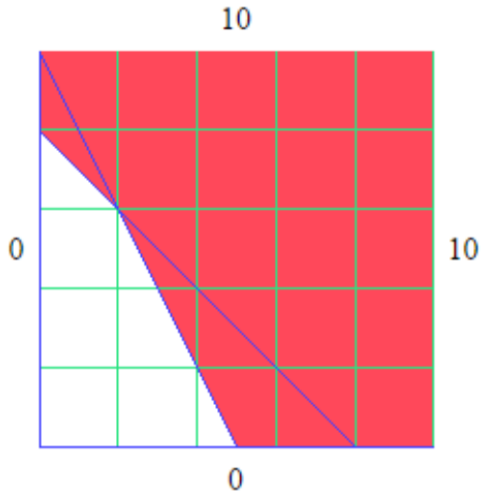

Рисунок 34: Зона допустимых решений в линейном программировании

Зона, отмеченная на Рисунок 344 белым, определяется заданными ограничениями и называется зоной допустимых решений. Наша задача – найти значения переменных, которые попадают в эту зону и при которых достигается максимальное значение целевой функции. Наиболее простой метод решения задачи, т.н. симплекс-метод [1], заключается в переборе всех вершин многогранника зоны допустимых решений, т.к. доказано, что решение всегда находится в одной из вершин. В данном случае решение — 2 гектара пшеницы и 6 гектаров кукурузы значение целевой функции — 28000 рублей.

Данный пример иллюстрирует задачи, решаемые в рамках парадигмы линейного программирования. Большинство приложений этой парадигмы для автоматической обработки языка, однако, опирается на более строгую формулировку задачи — целочисленное линейное программирование (или просто целочисленное программирование).

[1] Симплекс-метод - наиболее простой и наглядный способ решения задач ЛП. Он имеет экспоненциальную сложность и не подходит для решения задач с большим числом переменных.

В рамках целочисленного программирования для всех переменных вводится дополнительное ограничение: теперь переменные могут принимать только целые значения. Такое ограничение позволяет, в частности, делать переменные бинарными и моделировать логические ограничения в рамках оптимизируемой системы (т.н. булевское программирование). Введение булевых переменных позволяет использовать целочисленное программирование для задач принятия решений. В некоторых случаях задача целочисленного программирования может быть сведена к задаче линейного программирования с последующей проверкой решения на целочисленность, однако существуют более совершенные методы, которые используют ограничение на тип переменных для того чтобы найти решение быстрее (например, branch-and-bound, см. [Land, Doig, 1960]).

II.4.3 Модуль глобальной оптимизации

Итак, мы рассмотрели общие принципы линейного и целочисленного линейного программирование и можем теперь сформулировать задачу оптимизации, которая сделает результаты работы нашей системы приемлемыми с формальной точки зрения и в перспективе сможет повысить качество работы системы. Напомним, что требование, которому выход нашей системы должен удовлетворять, звучит следующим образом: каждая семантическая роль может быть приписана только одному узлу в предложении. Дополнительно мы хотели бы, чтобы такое приписание обеспечивало максимальную "уверенность" системы в предоставленных ответах. Мы будем считать, что уверенность системы при присвоении узлу того или иного класса пропорциональна весу этого класса, который в терминах метода опорных векторов определяется как расстояние от точки до разделяющей гиперплоскости. Веса в методе опорных векторов формируются

таким образом, что для опорного вектора вес всегда равен единице или минус единице (в зависимости от того, с какой стороны от разделяющей гиперплоскости он находится). Любая точка, получившая вес больше 1, может интерпретироваться как однозначно относящаяся к проверяемому классу, а любая точка с весом меньше -1 с точки зрения классификатора в этот класс не включена. Для точек в весом в интервале (-1,1) классификатор не может предложить уверенного решения, однако знак в любом случае свидетельствует о принадлежности точки к тому или иному классу.

Для того чтобы нашу задачу можно было решить с помощью целочисленного линейного программирования, необходимо сформулировать её в соответствующих терминах.

Пусть у нас имеется набор ролей $\{r_1, r_2, r_3 \dots r_i\}$ и набор узлов $\{n_1, n_2, n_3 \dots n_j\}$. Одна из ролей является "пустой" ролью и обозначает отсутствие семантической роли на выбранном узле. Эта роль имеет особый статус, т.к., в отличие от других ролей, может быть заполнена неограниченное число раз. Обозначим её как $r_\emptyset$. Введём набор переменных-индикаторов x_{ij}, которые обозначают, что роль r_i приписана узлу n_j, и набор переменных w_{ij}, в которых хранится вес роли r_iдля узла n_j, определённый классификатором на основе свойств, выделенных для данного узла. Следующий пример иллюстрирует семантику переменной :

Иван купил яблоко

	r_1(Покупатель)	r_2(Товар)	$r_\emptyset$
n_1(Иван)	$w_{11} = 0.8$	$w_{21} = 0.2$	$w_{\emptyset 1} = 0.1$
n_2(Яблоко)	$w_{12} = 0.4$	$w_{22} = 0.7$	$w_{\emptyset 2} = 0.2$

Таблица 4: Распределение весов в задаче ЛП

Задача оптимизации состоит в следующем. Необходимо выбрать значения переменных x_{ij} таким образом, чтобы максимизировать функцию уверенности $\sum_{i,j} x_{ij} w_{ij}$. При этом существует два принципиальных ограничения, которые должны быть соблюдены. Во-первых, одному узлу может быть приписана только одна роль. Для каждого узла n_j необходимо обеспечить $\forall j$: $\sum_i x_{ij} = 1$. За счёт того, что переменные x_{ij} целые и принимают значения $\{0,1\}$, это условие может быть верно только когда узел имеет единственную роль. Во-вторых, каждая роль, за исключением "пустой" роли, может быть приписана только один раз. Это условия выражается в терминах наших переменных следующим образом: $\forall i \neq \emptyset$: $\sum_j x_{ij} \leq 1$.

Обратим внимание на тот факт, что мы не требуем, чтобы каждая роль была выражена, а лишь настаиваем, чтобы каждая роль, если она выражена, была присвоена только один раз. Мы допускаем, что данное условие может быть сформулировано более гибко, т.к. некоторые роли имеют большую вероятность быть выраженными, чем другие. Если какая-то роль в обучающих данных почти всегда выражается, то можно было бы дополнительно повысить вероятность того, что она будет выявлена и на этапе применения классификатора. Тем не менее, с учётом того, что в русском языке часто встречается явление эллипсиса, а также с учётом неточностей в наших тренировочных и тестовых данных, мы считаем выбранную формулировку наиболее устойчивой к отклонениям.

Сформулированная таким образом задача оптимизации передаётся в модуль линейного программирования, который решает её с помощью симплексного метода. Ниже мы приводим пример полной формулировки задачи для предложения из Таблица 4:

Максимизировать:

$$x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22} + x_{\emptyset 1}w_{\emptyset 1} + x_{\emptyset 2}w_{\emptyset 2}$$

С учётом ограничений:

“каждый узел получает одну роль”

$$x_{11} + x_{21} + x_{\emptyset 1} = 1$$

$$x_{12} + x_{22} + x_{\emptyset 2} = 1$$

“каждая роль заполняется максимум один раз”

$$x_{11} + x_{12} \leq 1$$

$$x_{21} + x_{22} \leq 1$$

II.5 Особенности имплементация системы

В предудущих разделах мы описали механизм работы нашей системы. Прежде чем перейти к анализу результатов, кратко остановимся на деталях технической реализации системы. Кроме того, что такая информация необходима для воспроизведения полученных результатов, она также может быть полезна при создании альтернативных систем, основанных на тех же компонентах. Наконец, необходимо помнить, что в целом результат работы системы – это продукт взаимодействия нескольких многих факторов , а именно

- инструкций, в соответствии с которыми разметчики аннотировали корпус FrameBank
- токенизатора, использованного при подготовке материалов для разметки
- морфологического анализатора
- синтаксического анализатора
- качества и исходного материала кластеризации
- непосредственно нашей системы

Поскольку наша система находится в самом конце этой цепочки, результат её работы аккумулирует результаты, полученные на предыдущих этапах. В случае, когда разметка по семантическим ролям производится на основе синтаксическго корпуса, оказывается возможным оценить работу системы SRL изолированно, т.е. исходя из того, что автоматический анализ на предыдущих этапах был выполнен правильно. Поскольку в нашем распоряжении нет синтаксически аннотированного корпуса с разметкой по семантическим ролям, мы можем лишь оценить общее качество работы системы. Несмотря на то что такой подход хуже подходит для описания непосредственно работы системы, он, тем не менее, даёт лучшее

представление о результатах её работы в реальных условиях, когда данные перед поступлением в систему обрабатываются автоматически.

Ниже мы приводим технические детали предложенной системы автоматической разметки актантов. Напомним, что общая архитектура системы выглядит следующим образом:

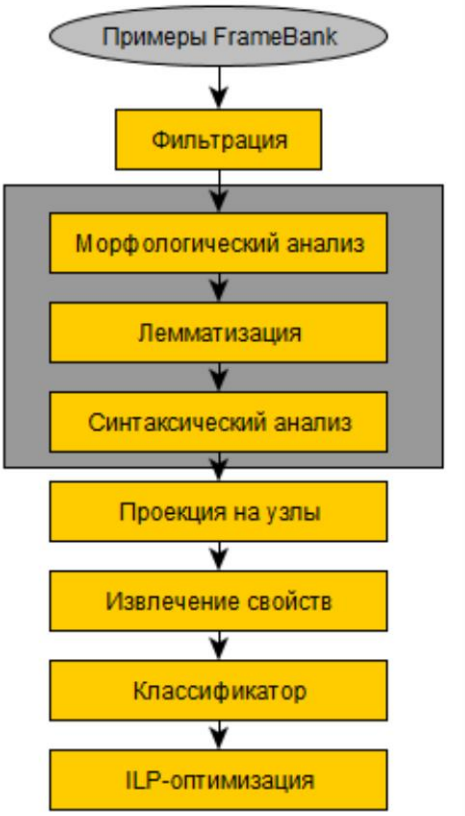

Рисунок 35: Архитектура системы

Исходные данные из корпуса FrameBank в формате xml проходят фильтрацию, после чего поступают в модуль предобработки. Фильтрация выполняется с помощью набора правил на языке python. В качестве модуля предобработки мы используем набор инструментов, созданный С. Шаровым и Й. Нивре [Sharoff, Nivre, 2011] и включающий в себя токенизатор, морфологический анализатор, лемматизатор и синтаксический парсер.

Поскольку исходная разметка корпуса FrameBank была выполнена на уровне токенов, т.е. отдельных слов, мы сохраняем разбиение на слова и предложения из исходного корпуса, чтобы оставить за собой возможность обогатить автоматические данные разметкой по семантическим ролям после завершения предварительной обработки. Тексты, разбитые на слова и предложения, передаются в модуль морфологического анализа на основе TreeTagger, который использует набор тегов MSD [Sharoff и др., 2008], разработанный для представления морфолгической информации для языков с богатой морфологией. TreeTagger был предложен в работе [Schmid, 1994] и является одним из наиболее популярных на сегодняшний день инструментов для морфологического анализа на основе машинного обучения. Анализатор основан на деревьях принятия решений и имплицитно снимает морфологическую неоднозначность, т.к. каждый лист дерева содержит только одну морфологическую метку. В TreeTagger используются тесты на полное лексическое совпадение (lex=дом) и тест на совпадение суффикса (*suffix=-ость*). Обучение производится на основе триграмм и таким образом информация о контексте слова также включается в модель. Следует отметить, что TreeTagger демонстрирует для различных языков качество, сопоставимое с моделями на основе вероятностей. В частности, модель, использованная в нашем эксперименте, демонстрирует точность (Accuracy) порядка 0.95 [Sharoff, Nivre, 2011].

Морфологический анализатор TreeTagger не только присваивает каждому слову морфологические показатели, но и определяет лемму слова в случае, если это слово содержалось в тренировочных данных, использованных для построения анализатора. В противном же случае возникает необходимость автоматически определить лемму неизвестного слова. Для этого в наборе инструментов, который мы используем, применяется лемматизатор CstLemma [Jongejan, Dalianis, 2009], который на основе суффиксов слов и

морфологической информации строит набор правил преобразования, позволяющих получить лемму для неизвестного слова. Авторы метода сообщают, что точность автоматической лемматизации на английском материале составляет порядка 0.97. Насколько нам известно, для русского языка подобного измерения не проводилось, однако ручной анализ результатов показывает, что лемматизация в подавляющем большинстве случаев выполняется правильно. Кроме того, отметим, что в рамках нашей задачи нам важна не столько правильность леммы, сколько совпадение лемм у соответствующих слов из тренировочной и тестовой выборки, что является более мягким условием.

Наконец, после автоматической морфологической обработки и лемматизации, данные попадают на вход синтаксического парсера. Мы используем модель MaltParser [Nivre, Hall, Nilsson, 2006], натренированную на основе корпуса СинТагРус. MaltParser представляет собой генератор итеративных синтаксических парсеров, которые последовательно определяют набор синтаксических зависимостей между словами в предложении на основе морфологической информации и леммы слова. Генератор, а также инструмент для подбора параметров модели MaltOptimizer [Ballesteros, Nivre, 2012], находятся в открытом доступе, и обучить модель с использованием альтернативного набора тегов не составляет труда, однако мы приняли решение использовать готовую модель для того, чтобы быть уверенными в совместимости наших компонентов предобработки. Синтаксический парсер, который мы используем, имеет качество порядка 82.3 LAS и 89.0 UAS [Sharoff, Nivre, 2011], что соответствует результатам, полученным для этого парсера на материале других языков.

После предварительной обработки данные передаются непосредственно в нашу систему классификации актантов. Система выполнена на языке программирования python. Для операций, связанных с машинным обучением,

мы используем библиотеку scikit-learn [Pedregosa и др., 2011], в частности, в роли классификатора используется имплементация LinearSVC с параметрами по умолчанию. Для сбора статистики и обработки данных была использована библиотека pandas [McKinney, 2011], которая предоставляет набор базовых операций для работы с данными.

Как уже указывалось, кластеризация в нашей имплементации выполняется с помощью инструмента Chinese Whispers, созданного авторами метода [Biemann, 2006b]. В качестве исходного векторного пространства мы использовали модель RusVectōrēs на основе новостного корпуса, предложенную А. Кутузовым [Kutuzov, Andreev, 2015]. Для работы с моделью и для извлечения списков семантически близких слов использовалась библиотека gensim, [Rehurek, Sojka, 2010]. В случае, когда мы строили кластеризацию только на основе имён существительных, определение части речи выполнялось с помощью анализатора pymorphy2[2].

Структуры и модули, ответственные за хранение и представление данных, за трансформацию форматов, за общий процесс обработки данных и за генерацию отчётов, были имплементированы на языке python. Для хранения информации о деревьях зависимостей, для поиска путей для свойства "путь" и извлечения списков потомков для проекции разметки FrameBank на зависимостные узлы был создан модуль представления деревьев зависимостей для языка python на основе библиотеки для работы с графами networkx[3].

Наконец, оптимизация на основе целочисленного программирования в нашей реализации системы выполняется с помощью библиотеки pulp [Mitchell, Sullivan, Dunning, 2011].

[2] https://pymorphy2.readthedocs.org/en/latest/

[3] https://networkx.github.io/

Как видно из описания, приведенного выше, предлагаемая имплементация достаточно неоднородна и объединяет в себе множество внешних компонентов. С практической точки зрения было бы удобно иметь цельную систему, которая кроме функционирования в исследовательском режиме с предобработанными тестовыми и тренировочными данными была бы также способна работать с произвольными входными данными. В нашем случае создание подобной системы было осложнено следующими факторами. Во-первых, основная логика нашей системы выполнена на языке python, предобработка же выполняется с помощью внешнего набора инструментов, который на практике представляет собой разрозненный набор скриптов и библиотек, интегрировать который в систему представляется затруднительным. Во-вторых, наша система не включает в себя модуля разрешения глагольной неоднозначности, однако в случае, если мы намереваемся работать с произвольными входными данными, глагольная неоднозначность должна быть снята, что в настоящий момент представляется трудновыполнимым. При условии решения упомянутых проблем создание полноценной системы, способной работать с произвольными данными, представляется вполне выполнимой задачей и было бы естественным продолжением описанной в данной диссертации исследовательской работы.

III. Экспериментальная оценка и результаты

III.1 Предмет и критерии оценки

Настоящая глава диссертации посвящена экспериментальной оценке результатов работы предложенной системы. Для задач машинного обучения эта область хорошо разработана и существует целый ряд стандартных параметров, по которым можно определить, насколько хорошо работает система. Поскольку мы сформулировали задачу автоматической обработки актантов как задачу классификации, речь далее пойдёт только о мерах качества, применимых к задачам классификации.

Задача оценки качества работы классификатора сводится к следующему. Предположим, что нам даны тренировочные данные, на которых система обучается. Эти данные разбиты на классы, и мы предполагаем, что для каждого класса существует некоторый абстрактный концепт, который управляет порождением экземпляров этого класса. Задача классификации – научиться распознавать этот концепт и отличать экземпляры данного класса от других на основании тренировочных данных. В то же время тренировочные данные

являются, как правило, далеко не исчерпывающими, т.к. для полного исчисления всех экземпляров, принадлежащих данному концепту, потребовалось бы исчислить все экземпляры вообще, что не представляется возможным. В этой связи мы обучаем классификатор на ограниченной по объёму выборке объектов, обучающей выборке, и чем более репрезентативной была эта выборка, тем в большей степени полученный классификатор будет соответствовать действительности.

Впрочем, установить, насколько концепт, полученный классификатором, соответствует реальному положению дел, также не представляется возможным по тем же самым причинам: для того, чтобы напрямую сравнить функцию, построенную классификатором, с целевой функцией, мы должны были бы иметь доступ к этой целевой функции (и если бы это было так, не было бы необходимости в обучении классификатора). Поэтому применяются различные методы аппроксимации, позволяющие приблизительно оценить качество работы классификатора и сравнивать различные классификаторы между собой. Можно было бы оценивать качество работы системы используя те же данные, на которых она была натренирована, однако подобная оценка зачастую является крайне неточной из-за **переобучения**. Следующий простой пример иллюстрирует это явление.

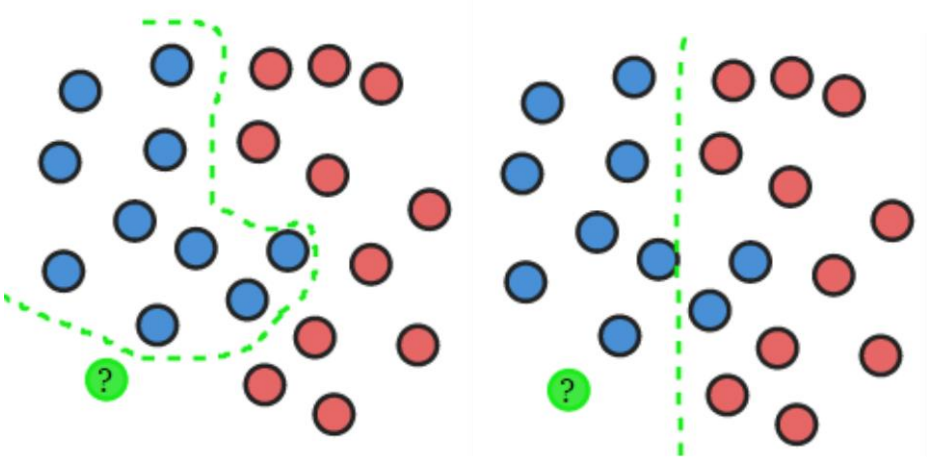

Рисунок 36: Проблема переобучения

Как мы видим, первая система идеально работает на тренировочных данных, однако обнаруженная ей закономерность слишком сложна и в результате новые данные оказываются классифицированы неверно. Концепт, выделенный второй системой, проще и хотя качество работы второго классификатора на тренировочной выборке ниже, он лучше справляется с классификацией новых экземпляров.

Существует несколько сценариев оценки качества систем с использованием новых данных. Наиболее тривиальной является оценка результатов работы системы экспертом. Исследователь вручную анализирует результаты работы системы и отмечает, в каких случаях система выдаёт верный ответ, а в каких – допускает ошибку. Этот подход является наиболее гибким, однако неэффективным в силу субъективности оценок эксперта и необходимости производить оценку каждый раз заново. Тем не менее, этот подход часто используется на ранних этапах разработки системы и позволяет делать выводы об особенностях работы алгоритмов, влиянии свойств и т.д.

Вторая популярная возможность – автоматическая оценка качества на тестовой выборке. Суть этого подхода состоит в том, что часть экземпляров изолируется и используется для тестирования системы в качестве нового, "неизвестного" материала. Этот подход позволяет объективно измерить качество работы системы, сравнивать системы между собой и быстро производить оценку качества работы при изменении параметров системы. Недостатки этого подхода состоят в расходовании аннотированных данных (которые можно было бы использовать для обучения) и чувствительности к составу тестовой выборки. Для борьбы с этим используется случайный выбор экземпляров и кросс-валидация, при которой данные разделяются на несколько фрагментов, каждый из которых используется для тестирования системы, натренированной на оставшихся данных. В качестве окончательного показателя используется усредненное качество работы системы по этим

фрагментам. Наконец, последний вариант, используемый в прикладных системах – проверка качества системы на этапе применения. Этот способ подходит только для реально используемых систем и в научных работах применяется редко.

В рамках данного исследования мы будем использовать оценку качества на основании тестовой выборки. В наиболее интересных случаях мы прибегаем к экспертному анализу результатов.

Существует несколько способов оценки качества работы классификатора с использованием тестовой выборки. Ключевым элементом оценки качества является **матрица ошибок** (*confusion matrix*), которая отражает количество правильных и неправильных срабатываний классификатора на тестовых данных. В случае бинарной классификации матрица выглядит следующим образом:

predicted\actual	+	-
+	True positive (TP)	False positive (FP)
-	False negative (FN)	True negative (TN)

Таблица 5: Структура матрицы ошибок

Матрица может быть расширена и для общего случая, где число классов больше 2.

Наиболее примитивный метод оценки качества классификации заключается в подсчёте числа правильных срабатываний классификатора относительно общего числа срабатываний, или *true positive rate* (*TPR*):

$$TPR = TP/(TP + FN)$$

В паре с этим показателем используется показатель *false negative rate* (*FNR*):

$$FNR = FN/(TP + FN)$$

Гибридом этих двух показателей является мера **аккуратности**[4] (*Accuracy*, *ACC*), которая вычисляется для случая бинарной классификации как

$$ACC = (TP + TN)/(TP + FP + TN + FN)$$

Эта мера отражает общее качество работы классификатора и часто используется в исследованиях по машинному обучению. Тем не менее, для задач нашего исследования она подходит не очень хорошо. Как мы помним, наша формулировка задачи SRL подразумевает классификацию экземпляров по семантическим ролям. В случае отсутствия роли экземпляр получает специальныую метку класса None. Это так называемый класс большинства, и качество работы классификатора, определяющего этот класс, будет в значительной степени определять наш показатель аккуратности. Представим себе следующую матрицу ошибок:

predicted\actual	+	-
+	TP = 2	FP = 2
-	FN = 5	TN = 100

Таблица 6: Пример матрицы ошибок

Как мы можем видеть, при расчёте аккуратности качество работы малочисленного положительного класса фактически нивелируется качеством работы негативного класса-большинства. В результате этого при изменении классификатора изменения аккуратности (при условии, что класс-большинство выделяется стабильно хорошо) будут незначительными, и система, которая не выделяет семантических ролей вообще, окажется вполне

[4] Мы намеренно не используем здесь термин *точность*, чтобы избежать неоднозначности: точностью в данной работе мы называем меру *precision*

конкурентоспособной с точки зрения этой метрики, что не соответствует нашим намерениям.

Для того чтобы избежать проблем, связанных с классом-большинством, используются меры **точность** (*Precision*, *P*) и **полнота** (*Recall*, *R*). Точность показывает долю правильно классифицированных объектов данного класса в общей выдаче системы. Полнота отражает долю объектов выбранного класса, обраруженных системой. **Мера F1** представляет собой гармоническое среднее точности и полноты и призвана объединить эти два показателя в один таким образом, чтобы невозможно было завысить качество работы системы путём завышения точности в ущерб полноте и наоборот (что произошло бы, если бы мы использовали, например, среднее арифметическое точности и полноты):

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2PR}{P + R}$$

Эти меры вычисляются отдельно для каждого класса, и затем полученные показатели усредняются по всем классам. Такое усреднение называется макро-средним и противопоставляется микро-среднему, при котором число правильных и неправильных срабатываний системы сначала суммируется по всем классам, а затем на основании этих величин вычисляются показатели качества. Таким образом, при использовании макро-усреднения влияние класса-большинства на общую оценку качества работы классификатора уменьшается пропорционально количеству классов.

Вопрос о том, как именно следует оценивать качество работы систем автоматической разметки актантов, является нетривиальным. В ряде работ, посвящённых проблеме автоматического выделения семантических ролей, разметка актантов производится в два этапа, каждый из которых подвергается

отдельной оценке. На первом этапе вычисляется качество работы классификатора, определяющего, является ли узел или отрезок текста актантом или нет. Затем на основании этих данных определяется качество распределения актантов по ролям. Поскольку в нашем случае эти операции выполняются одновременно, мы считаем, что выбранный нами метод оценки является объективным и соответствует принятым в области машинного обучения стандартам. Качество класса-большинства, несмотря на взвешивание оценок, всё ещё оказывает большое влияние на конечный результат классификации, однако в нашей формулировке задачи способность классификатора правильно присваивать метку отсутствия роли не менее важна, чем способность присваивать ту или иную семантическую роль. При сравнении результатов работы различных конфигураций классификатора мы приводим также данные для "наивной" системы, которая всегда выбирает класс-большинство (т.е. в нашем случае класс None). Это позволяет составить представление о качестве, получаемом при использовании наиболее простой стратегии, а также оценить вклад класса-большинства в общее качество работы системы.

III.2 Процедура оценки

Как уже упоминалось ранее, при оценке работы системы мы преследуем две цели. Во-первых, с помощью метрик качества мы хотим определить общее качество работы системы на тестовых данных. Несмотря на то, что оценка, полученная таким образом, не является абсолютно точной, она позволяет получить хотя бы приблизительное непредвзятое представление о том, насколько хорошо система выполняет поставленную задачу, а также в перспективе позволяет сравнивать системы между собой. Во-вторых, в ходе оценки мы хотим определить вклад отдельных свойств, а также иных параметров системы в качество классификации и затем, путём экспертного анализа, определить достоинства и недостатки отдельных свойств и параметров, от значений которых зависит конкретная конфигурация системы и результаты её работы. Остановимся на этих параметрах подробнее. Первый и наиболее очевидный из них – это набор свойств, используемых системой. Всего в системе представлено девять свойств, условно разделённых на две группы: семантические и синтаксические. С учётом модификаций свойств "путь" и "кластер" список выглядит следующим образом:

- синтаксические свойства
    - путь (*path*)
    - короткий путь (*shortPath4*)
    - падеж (*case*)
    - финский падеж (*finncase*)
    - форма глагола (*vform*)
    - залог (*voice*)
- семантические свойства
    - лемма (*lemma*)

- кластер (*cluster*)
- часть речи (*POS*)

Каждое свойство по отдельности делает вклад в качество работы классификатора, и наиболее тривиальным способом оценить важность каждого из свойств было бы произвести обучение с использованием только этого свойства и сравнить результаты. Сложность, однако, состоит в том, что некоторые свойства адекватно описывают класс только в комбинации, в результате чего изолированное тестирование свойств не дает возможности полностью оценить их значимость. В связи с этим в нашей работе мы анализируем качество работы классификатора для всех возможных комбинаций свойств. Использование такого подхода позволяет оценить качество и вклад каждой комбинации и определить, какие свойства хорошо работают в связке, а какие при комбинировании мешают классификатору построить адекватную целевую функцию. Всего при таком подходе необходимо протестировать 2^9 комбинаций.

Еще одним параметром, влияющим на качество работы системы, является метод кластеризации, который используется при порождении свойства "кластер". При построении кластеров мы используем две различных конфигурации: в первом случае при создании графа используются только имена существительные, во втором – все слова, доступные в исходной модели. Данный параметр релевантен только в случаях, когда свойство "кластер" включено в набор для тестирования.

Поскольку добавление каждого бинарного параметра удваивает число экспериментов, которые необходимо провести, было принято решение производить оценку системы в два этапа. Тестирование качества работы с использованием описанных выше конфигураций составляет первый этап оценки системы, на котором основной целью является оценка вклада отдельных свойств и их комбинаций в общее качество работы классификатора.

По результатам первого этапа были выбраны пять лучших конфигураций системы в терминах F-меры для каждой из трёх групп свойств: синтаксических, семантических и их комбинации. Полученные пятнадцать лучших конфигураций были протестированы более детально на втором этапе оценки.

На втором этапе оценивался вклад модуля постобработки на основе линейного программирования в качество работы системы. Действительно, несмотря на то, что вывод нашей системы без участия этого модуля формально некорректен, это не мешает нам оценивать её качество с помощью выбранных метрик. Тем не менее, кажется разумным предположить, что использование модуля постобработки не только приводит вывод системы в соответствие с формальными требованиями, но и может повысить качество работы системы за счёт дополнительной оптимизации результатов на уровне предложения (до этого момента система работает только на уровне отдельных узлов в дереве зависимостей).

Кроме того, на втором этапе оценивается влияние частотного порога и соотношения размеров тренировочной и тестовой выборок на результат классификации.

Рассмотрим более подробно то, каким именно образом рассчитывались значения метрик качества для каждой из приведённых конфигураций. В соответствии с выбранным методом машинного обучения, в процессе работы система тренирует множество классификаторов типа “*один против всех*”, каждый из которых должен быть оценен в терминах точности, полноты и F-меры. Поскольку такой большой объём данных интерпретировать трудно, были использованы усреднённые меры, процедура расчёта которых, несмотря на свою простоту, нуждается в эксплицитном описании.

Предложения исходного корпуса были сгруппированы в зависимости от того, какую конструкцию они описывают. Каждый из полученных подкорпусов в свою очередь случайным образом разбивается на тестовую и тренировочную

выборки. Затем, с использованием тренировочной выборки производится обучение классификаторов типа "один против всех" для каждой роли. При использовании системы на тестовых данных каждый экземпляр передаётся каждому из бинарных классификаторов, которые, в свою очередь, возвращают вес – меру "уверенности" классификатра в том, что данный экземпляр принадлежит к его классу. Затем система выбирает класс с наибольшим весом и приписывает его экземпляру. В случае с ILP-постобработкой выбор классов производится путём решения LP-задачи оптимизации и максимизирует суммарную "уверенность" классификатора на всём предложении. В любом случае, в результате применения системы каждый экземпляр (узел дерева зависимостей) входного предложения получает одну из ролевых меток. На основании этих данных *для каждой роли* рассчитывается точность, полнота и F-мера. Эти меры усредняются по всем ролям, и полученный результат считается результатом работы системы для выбранной конструкции. Затем значения мер усредняются ещё раз по всем конструкциям, и полученные средние для точности, полноты и F-меры в дальнейшем считаются "качеством работы системы в данной конфигурации". Подобная группировка, хотя может на первый взгляд показаться излишней, позволяет снизить влияние класса-большинства (в случае с усреднением по ролям) и частотных конструкций (в случае с усреднением по конструкциям) на общий результат. Кроме того, при такой группировке мы можем вычислить не только среднее, но и стандартное отклонение мер качества по конструкциям, что позволяет оценить разброс значений метрик для различных конструкций.

Поскольку для работы системы необходимо установить все параметры, на первом этапе оценки мы используем значения частотного фильтра и соотношения тренировочной и тестовой выборок по умолчанию (минимальная частота 20 и соотношение выборок 40/60 соответственно) и не используем ILP-

оптимизацию. Следующая схема демонстрирует структуру нашей процедуры оценки в графическом виде:

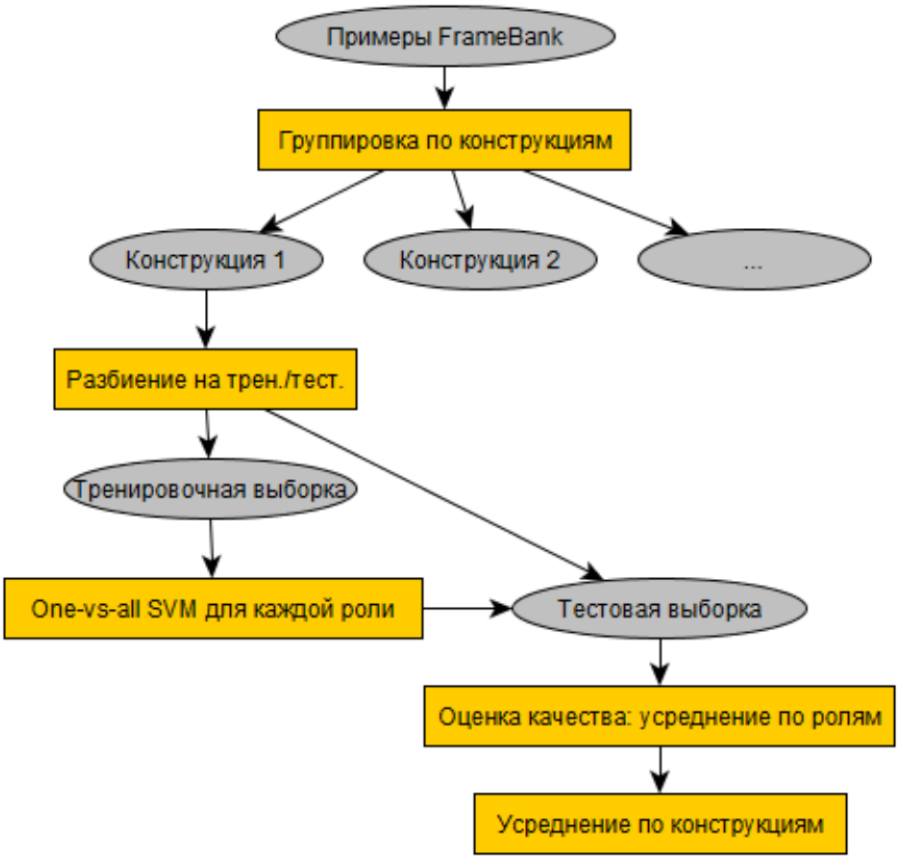

Рисунок 37: Процедура оценки качества

III.3 Результаты

Данный раздел представляет собой краткий обзор результатов работы системы, полученных с помощью описанной ранее процедуры оценки. Следует подчеркнуть тот факт, что наша процедура оценки отличается от наиболее распространённой в области автоматической классификации актантов. Так, большинство исследователей оценивают свои системы на основе качества выделения собственно ролевых показателей. Другими словами, при оценке рассчитывается точность, полнота и F-мера только для узлов, содержащих метку роли, отрицательный класс же, т.е. токены, которые не содержат ролевой метки как в экспертной, так и в автоматической разметке, игнорируется. Мы полагаем, что выбор данной процедуры оценки связан с традиционным делением задачи на идентификацию актантов и классификацию актантов. При такой постановке задачи отдельно измеряется качество идентификации, а затем, зачастую исходя из того, что идентификация была выполнена верно, т.е. на "идеальных" актантах, производится оценка правильности распределения актантов по ролям. В то время как при такой постановке задачи оценка только на основании ролевых узлов может быть оправданной, в случае, когда идентификация и классификация актантов выполняется одновременно, кажется более правильным включать в оценку и "пустой" класс None. Действительно, этот класс является наиболее многочисленным, однако с точки зрения задачи классификации он ничем не отличается от остальных классов, и потому, на наш взгляд, неправильно исключать его из сравнения, несмотря на то, что это может привести к некоторому "завышению" показателей по сравнению с традиционным методом.

С целью создать у читателя объективное представление о качестве работы нашей системы, при изложении результатов первого этапа оценки мы дополнительно приводим результат базовой системы (*baseline*), которая голосует за класс большинства.

Данный раздел состоит из двух частей, соответствующих двум этапам оценки качества, описанным выше. Задача первого этапа оценки – определить, использование каких свойств и комбинаций свойств даёт наилучшие результаты с точки зрения качества работы системы. Задача второго этапа – оценить влияние ограничения на частоту конструкции, соотношения размеров тренировочной и тестовой выборок, а также эффект от ILP-оптимизации. Следует отметить, что одно из отрицательных свойств классификаторов на основе методов опорных векторов – сложность интерпретации результатов и невозможность визуализировать модель в случае многоразмерных данных. При оценки влияния конкретных параметров мы будем опираться на коэффициенты корреляции значений параметров с итоговыми показателями качества. Напомним, что наша система имеет следующие параметры, которые мы будем варьировать в рамках экспериментов:

- features – набор свойств
- tts – соотношение тренировочной и тестовой выборок, а именно, доля тестовых данных в общем объёме
- thr – частотная граница отсечения конструкций
- ilp – использование ILP
- cluster – метод кластеризации

Результат оценки качества работы системы первого этапа будут дополнительно сопоставлены с значениями для "традиционных" метрик, а также с результат "базовой" системы, которая всегда голосует за класс большинства (класс None).

III.3.1 Влияние свойств на классификацию узлов

Для оценки вклада индивидуальных свойств и их комбинаций в качество работы системы остальные параметры системы были зафиксированы на значениях по умолчанию: при этом не производится ILP-оптимизация, частотный фильтр на конструкции устанавливается равным 20, соотношение тестовой и тренировочной выборок составляет 40/60.

Система была запущена на каждой из возможных комбинаций свойств и типов кластеров (в случаях, когда свойство "кластер" используется), результаты были усреднены по конструкциям в соответствии с описанной ранее процедурой, и для каждой из комбинаций были получены средние значения а также квадратные отклонения (*standard deviation*, *std*) для метрик P, R, F1, Acc, а также для метрик на основе ролей (role-P, role-R, role-F, role-Acc). Всего на этом этапе было проанализировано 1535 конфигураций системы. Мы сгруппировали результаты по использованным в конфигурациях классам свойств в соответствии с описанным ранее разделением на семантические и синтаксические свойства. Ниже в Табл. 7-9 приводятся пять лучших конфигураций системы, основанных только на синтаксических, только на семантических и на полных наборах свойств.

Все свойства

Features	P	R	F	Acc
Voice,POS,finncase,prep_lemma,case,shortPath4	0.759	0.667	0.695	0.950
Vform,prep_lemma,case,shortPath4	0.765	0.666	0.694	0.951
Vform,finncase,prep_lemma,case,shortPath4	0.764	0.667	0.694	0.950
Voice,vform,POS,finncase,lemma,prep_lemma,path,case,shortPath4	0.754	0.668	0.694	0.950
Vform,finncase,prep_lemma,path,case,shortPath4	0.761	0.667	0.694	0.950
baseline	0.331	0.356	0.343	0.928

Таблица 7: Лучшие конфигурации системы, все свойства;

здесь и далее приводятся средние значения

Только синтаксические свойства

Features	P	R	F	Acc
Vform,prep_lemma,case,shortPath4	0.765	0.666	0.694	0.951
Vform,finncase,prep_lemma,case,shortPath4	0.764	0.667	0.694	0.950
Vform,finncase,prep_lemma,path,case,shortPath4	0.761	0.667	0.694	0.950
Vform,prep_lemma,path,case,shortPath4	0.761	0.667	0.693	0.950
Voice,vform,prep_lemma,case,shortPath4	0.762	0.666	0.692	0.950
baseline	0.331	0.356	0.343	0.928

Таблица 8: Лучшие конфигурации системы, синтаксические свосйтва

Только семантические свойства

Features	P	R	F	Acc
POS,lemma,cluster-all	0.514	0.424	0.433	0.925
POS,lemma,cluster-nouns	0.514	0.424	0.433	0.925
Lemma,cluster-all	0.513	0.422	0.431	0.925
Lemma,cluster-nouns	0.513	0.422	0.431	0.925
POS,lemma	0.521	0.420	0.429	0.926
baseline	0.331	0.356	0.343	0.928

Таблица 9: Лучшие конфигурации системы, семантические свойства

Как мы можем наблюдать, формально наилучшие показатели качества достигаются при использовании семантических и синтаксических свойств, однако использование только синтаксических свойств позволяет добиться схожих результатов. В то же время системы, основанные только на семантических свойствах, демонстрируют значительно худшее качество работы, впрочем, всё равно превосходя по качеству наш базовый классификатор.

Следующие графики позволяют оценить профиль результатов работы нашей системы отдельно для показателей точности, полноты и F-меры. По оси x расположены системы в порядке убывания F-меры, а по оси y – значения соответствующей метрики. Как можно видеть из графика на Рис. 38, качество систем остаётся достаточно стабильным и варьируется лишь незначительно до определённого момента, а затем резко падает. Этот момент соответствует отключению свойств, связанных с путём в дереве зависимостей. В целом можно отметить, что система имеет приоритет точности над полнотой.

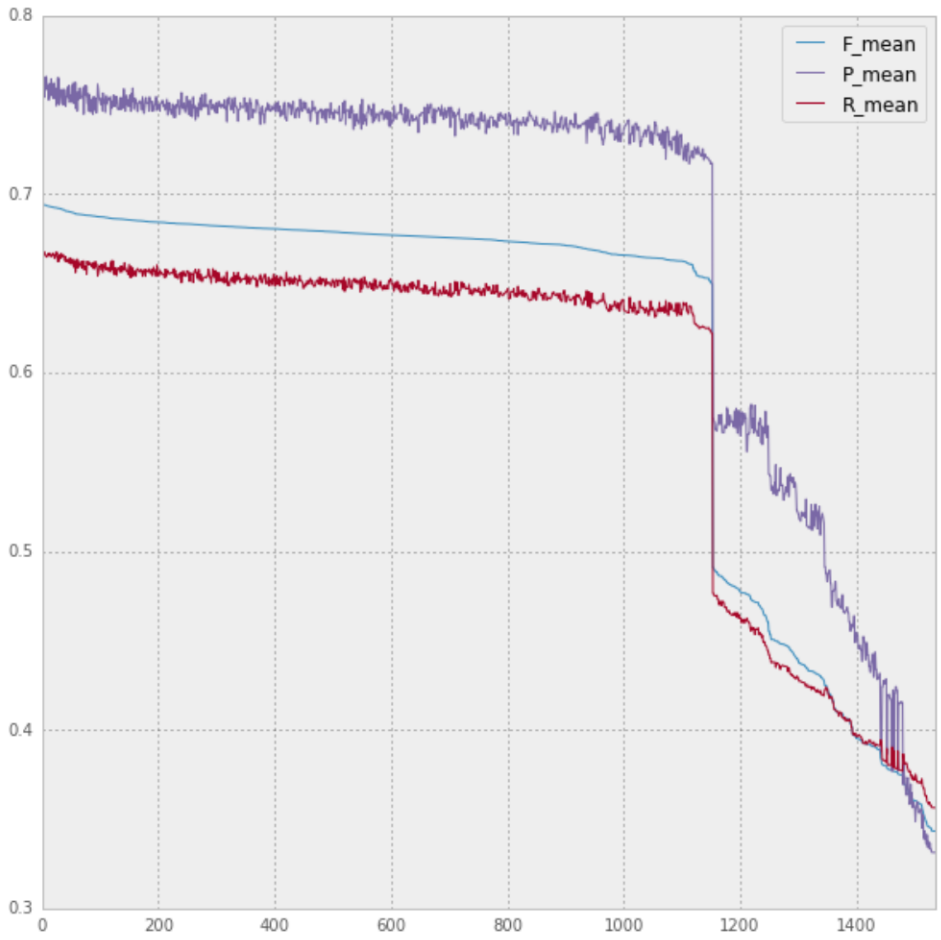

Рисунок 38: Профиль конфигураций системы

Ниже также приводится график квадратичных отклонений для значений точности, полноты и F-меры (см. Рис. 39). Как видно из графика, исключение свойств, связанных с синтаксическим путем, не только приводит к падению

точности и полноты системы, но и влечет за собой значительный разброс показателей точности по конструкциям.

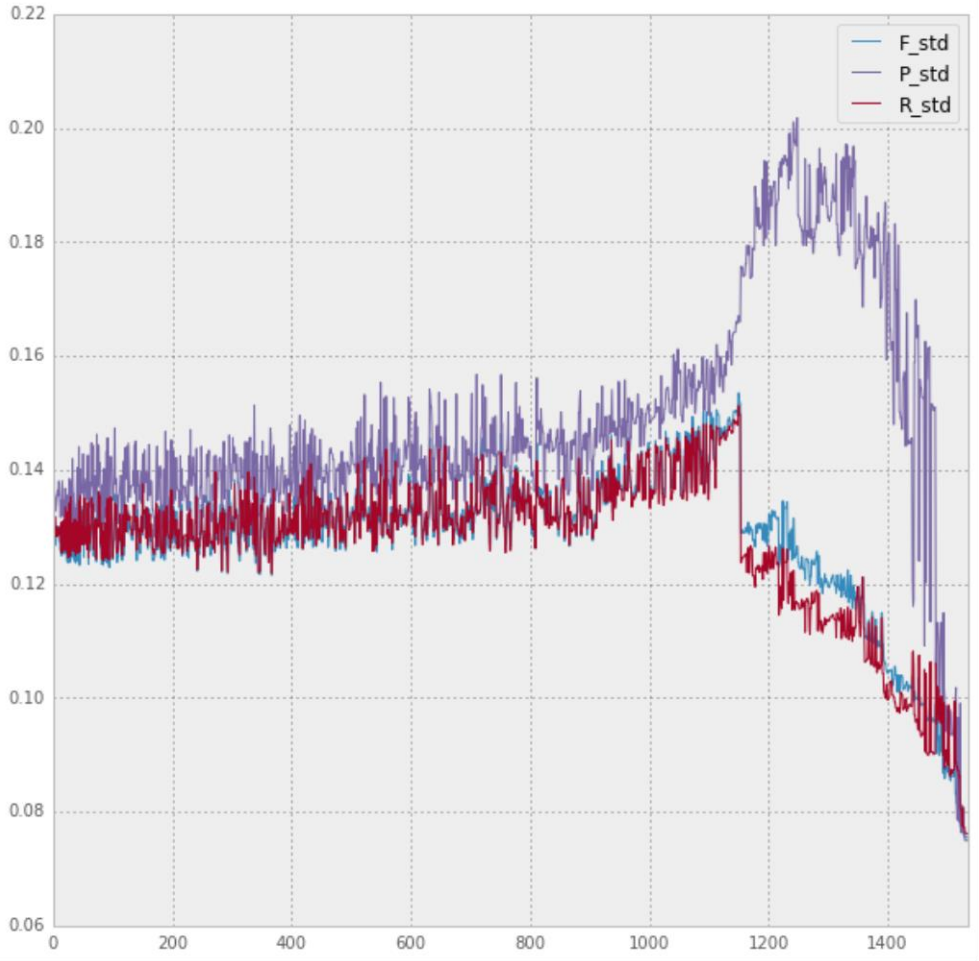

Рисунок 39: Профиль квадратичных отклонений

Наиболее успешными являются системы, натренированные на комбинациях из полного и синтаксического наборов. Отметим, однако, что наилучший результат достигается при использовании лишь подмножества всех доступных свойств. Для того чтобы оценить вклад отдельных свойств в качество работы классификатора, был рассчитан показатель корреляции каждого из этих свойств с показателями качества. Соответствующая таблица приводится ниже:

Свойство	F	P	R
POS	0.024	0.020	0.026
Lemma	0.094	0.148	0.075
Cluster	-0.005	0.022	-0.015
Case	0.055	0.059	0.060
Finncase	0.008	0.011	0.007
Vform	0.006	0.007	0.008
Prep_lemma	0.025	0.026	0.026
Voice	-0.004	-0.010	-0.002
Path	0.572	0.551	0.576
ShortPath4	0.574	0.543	0.583

Таблица 10: Корреляция свойств с качеством работы системы

Как видно из Табл. 10, наиболее высокой корреляцией среди индивидуальных свойств обладают свойства "короткий путь" и "путь". Небольшой положительный вклад вносят также свойства "лемма", "падеж", "финский падеж", "часть речи" и "предложная лемма". Влияние свойства "кластер" на F-меру крайне незначительно. Также мы проанализировали вклад в качество работы классификатора комбинаций свойств. Ниже мы приводим данные по коэффициенту корреляции единичных свойств и пар с показателями качества классификатора (приведены конфигурации с наиболее высоким коэффициеном корреляции):

Свойство	F	P	R
ShortPath4	0.574	0.543	0.583
Path	0.572	0.551	0.576
Cluster+shortPath4	0.389	0.372	0.392
Cluster+path	0.389	0.378	0.387
Case+shortPath	0.350	0.330	0.359
Case+path	0.349	0.335	0.353
POS+shortPath4	0.347	0.327	0.353
POS+path	0.346	0.331	0.349
Prep_lemma+shortPath4	0.346	0.327	0.351
Prep_lemma+path	0.345	0.332	0.347

Таблица 11: Корреляция пар свойств с качеством работы системы

Как мы можем видеть, вклад свойств, основанных на пути, всё ещё очень велик, однако полезными оказываются и комбинации этих свойств с информацией о кластере и частеречной принадлежности слова.

Ниже мы еще остановимся на вопросах, связанных с влиянием свойства "путь" и незначительностью вклада кластеризации в итоговый результат работы системы, а сейчас перейдём к описанию результатов второго этапа оценки качества, в ходе которого мы определили вклад ILP-оптимизации, а также влияние размеров выборки и соотношения объёмов тренировочных и тестовых данных на качество классификации.

III.3.2 Влияние глобальной оптимизации, размера тестовой выбоки и ограничения на частоту конструкции

На данном этапе пять лучших конфигураций свойств для каждого из классов свойств были проанализированы с точки зрения влияния ILP-

оптимизации, ограничения на частоту встречаемости конструкции и соотношения тестовой и тренировочной выборок на итоговое качество работы системы. В ходе тестирования измерялось качество системы на следующих комбинациях параметров

- ILP: да/нет
- Размер тестовой выборки (tts): 0.1, 0.2, 0.3, 0.4, 0.5
- Порог отсечения по количеству примеров (thr): 10, 20, 30, 40

Всего было протестировано 600 конфигураций системы. Топ-10 наилучших результатов приводится в следующей Табл. 12:

Features + ILP + thr + tts	P	R	F	Acc
vform,finncase,prep_lemma,path,case,shortPath4__ilp_False__thr_40__tts_0.1	0.799	0.744	0.760	0.958
vform,finncase,path,case,shortPath4__ilp_False__thr_40__tts_0.1	0.796	0.744	0.758	0.957
vform,path,case,shortPath4__ilp_False__thr_40__tts_0.1	0.796	0.743	0.757	0.957
vform,finncase,prep_lemma,path,case,shortPath4__ilp_True__thr_40__tts_0.1	0.796	0.733	0.755	0.958
vform,path,case,shortPath4__ilp_True__thr_40__tts_0.1	0.794	0.731	0.752	0.957
vform,finncase,prep_lemma,case,shortPath4__ilp_True__thr_40__tts_0.1	0.799	0.731	0.752	0.959
voice,vform,finncase,path,case,shortPath4__ilp_False__thr_40__tts_0.1	0.793	0.736	0.751	0.956
vform,finncase,prep_lemma,case,shortPath4__ilp_False__thr_40__tts_0.1	0.796	0.735	0.751	0.958
vform,prep_lemma,case,shortPath4__ilp_False__thr_40__tts_0.1	0.795	0.735	0.750	0.957
vform,prep_lemma,case,shortPath4__ilp_True__thr_40__tts_0.1	0.797	0.729	0.750	0.958

Таблица 12: Результаты глобальной оптимизации и влияние размера тестовой выборки.

Как видно из таблицы, ожидаемо лучшие результаты демонстрируют системы, для которых доступно наибольшее количество тренировочных данных, с набором свойств, показавшим также лучшие результаты на первом этапе тестирования. Несколько неожиданным является тот факт, что ILP-

оптимизация не всегда приводит к повышению качества в терминах F-меры. Для того, чтобы более наглядно проиллюстрировать эффект ILP-оптимизации, обратимся к следующим графикам, демонстрирующим профиль качества работы системы при включённой и отключённой ILP-оптимизации соответственно.

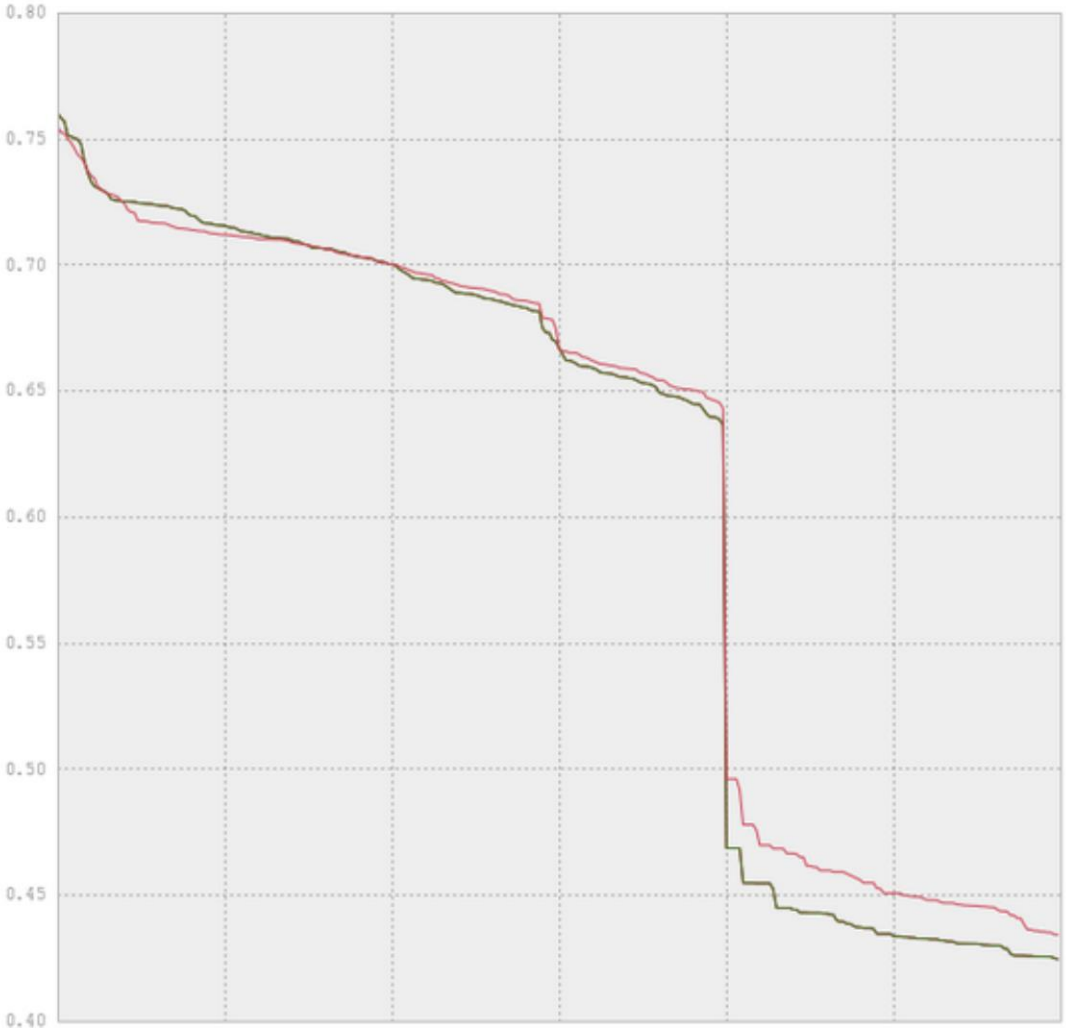

Рисунок 40: Профиль конфигураций системы с (красный цвет) и без (зелёный цвет) глобальной оптимизации

Как мы можем видеть, в целом ILP-оптимизация оказывает незначительный эффект на качество работы системы. В этой связи, однако, хочется отметить следующее обстоятельство. Для того, чтобы вывод системы был формально верным, нам *необходимо* использовать ILP-оптимизацию или

другой механизм, который обеспечит единственность заполнения каждой роли. В этом контексте тот факт, что ILP-оптимизация не оказывает негативного влияния на качество, а в некоторых случаях улучшает его, является безусловно положительным обстоятельством. Так, например, мы можем видеть, что ILP-оптимизация сглаживает падение качества при отказе от свойств на основе пути, которое мы уже наблюдали ранее.

Теперь обратимся к зависимости качества работы от ограничения на частоту встречаемости конструкции. Эту зависимость иллюстрирует следующий график:

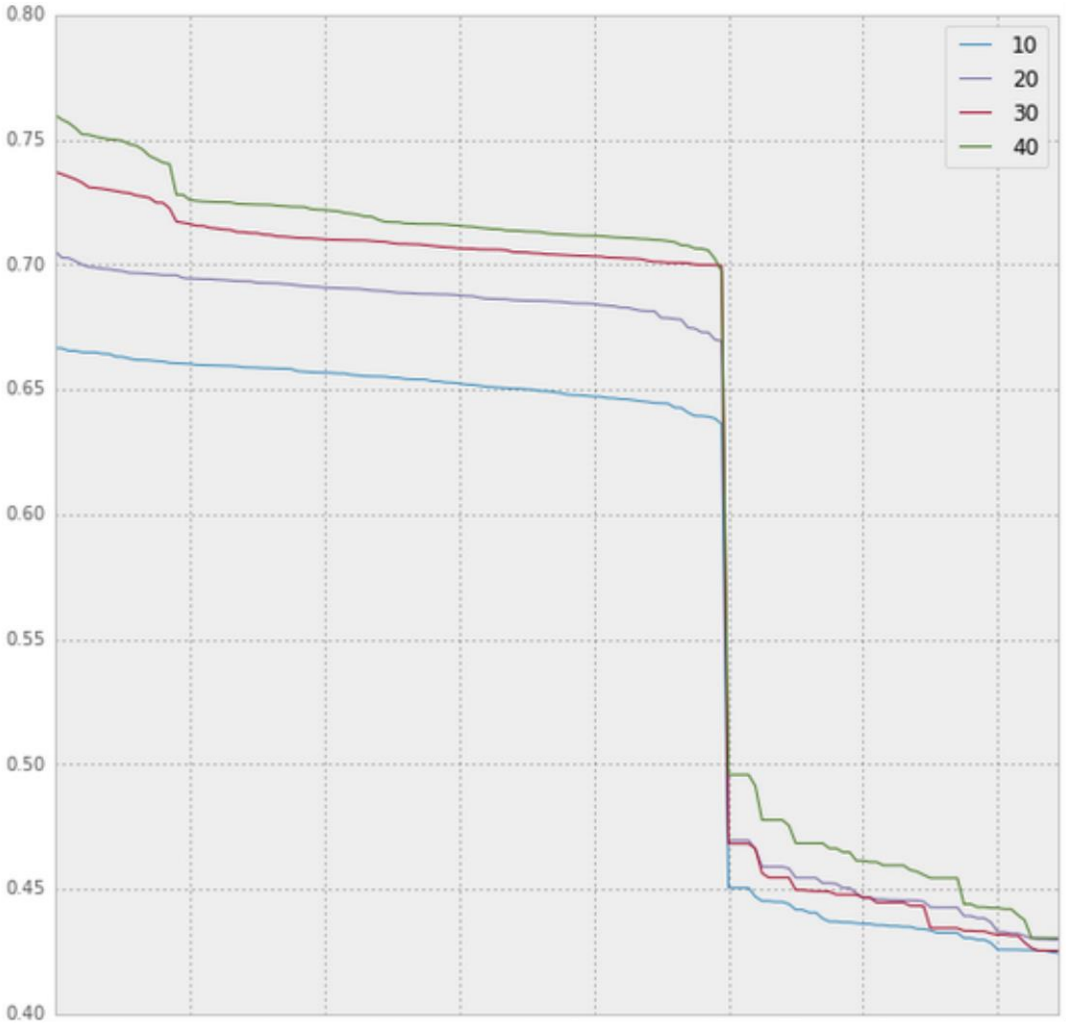

Рисунок 41: Зависимость качества работы системы от ограничения на частоту конструкции

Как мы видим, конструкции, для которых доступно большее число данных, стабильно демонстрируют лучшее качество работы системы, что вполне ожидаемо.

Наконец, рассмотрим зависимость качества работы системы от соотношения объёмов тестовой и тренировочной выборок.

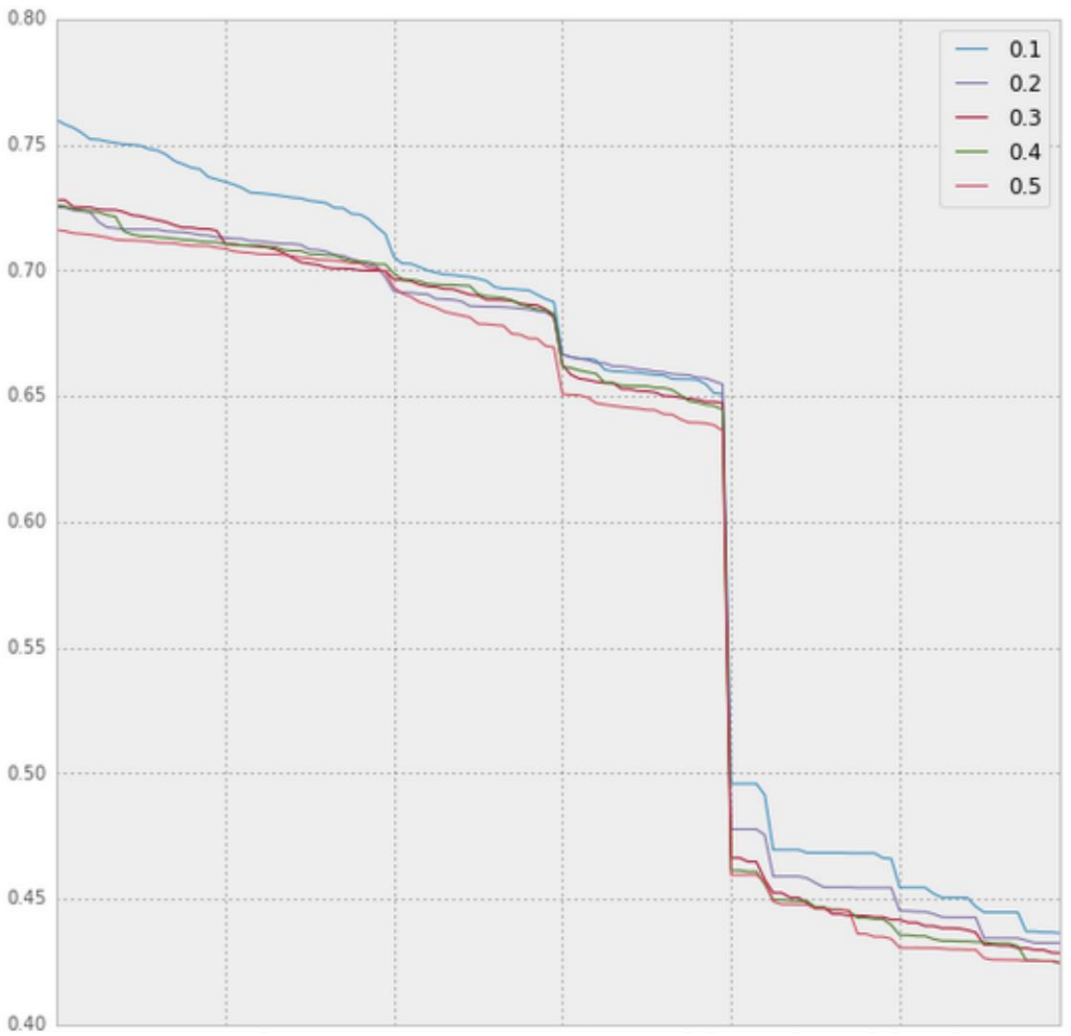

Рисунок 42: Зависимость качества работы системы от размера тестовой выборки

Мы можем наблюдать, что качество работы систем с большим объёмом тренировочных данных ожидаемо выше независимо от ограничения на частоту встречаемости конструкции. Отметим, однако, что разница по качеству между долей тестовых данных 0.2-0.5 невелика. Это явление представляет определённый интерес. По-видимому, объяснением в данном случае является некоторое количество неточностей в наших тренировочных и тестовых данных: в случае, когда тестовые данные содержат ошибки разметки, иногда классификатору оказывается "выгодно" иметь большее количество тестовых данных, т.к. таким образом уменьшается процент случайных ошибок, связанных с разметкой и проекцией разметки на синтаксические узлы. В качестве иллюстрации рассмотрим следующие примеры из тестового подкорпуса для конструкции “исчезнуть 1.1”:

A. Роль региональных лидеров при Путине резко снизилась , политические партии тоже резко пошли на убыль , **СМИ[X]** вообще **исчезли** .

B. Красивый **люд[X]** давно **исчез** .

C. Вся **группа[X]** , бесшумно скрывшаяся в непроглядной темени , пропала , **исчезла** , и следы её замыл дождь .

D. Вот **чай[X]** хороший совсем **исчез** .

Пример 12: Случайные ошибки как причина повышения качества с увеличением тестовой выборки. Красным цветом отмечены случаи, в которых, согласно алгоритму оценки, система приписала неправильную роль

Как можно видеть из Пример 12: Случайные ошибки как причина повышения качества с увеличением тестовой выборки12, система часто срабатывает правильно, однако в случаях C и D результат системы, хотя и правильный, не совпадает с экспертной разметкой. В случае небольших тестовых выборок повышается вероятность того, что доля предложений с

ошибочной разметкой в тестовых данных возрастёт, а формальное качество работы системы, обучившейся правильному концепту, понизится.

В завершение описания результатов работы системы кажется уместным привести матрицу корреляции значений tts, thr и ilp с качеством работы системы, так же, как мы делали это для отдельных свойств и их комбинаций на первом этапе:

	F	**P**	**R**
ILP	0.025	0.021	0.012
thr	0.163	0.225	0.164
tts	-0.058	0.003	-0.079

Таблица 13: Корреляция параметров с качеством работы системы

Значения коэффициентов корреляции показывают, что ILP-оптимизация оказывает незначительный положительный эффект на качество. Качество также растёт с увеличением ограничения на частоту конструкции и незначительно падает при увеличении доли тестовой выборки в общем объёме данных.

III.4 Обсуждение результатов

Итак, результаты оценки системы позволяют нам сделать следующие выводы:

1. Наилучшие результаты достигаются при использовании комбинированных семантико-синтаксических наборов свойств, однако и синтаксических свойств зачастую оказывается достаточно для достижения качества, близкого к максимальному. Особое значение имеет свойство “путь”, которое во многом определяет результат классификации в случаях, когда оно включено в признаковый набор. При этом ограничение длины пути оказывает положительный эффект на качество классификации.

2. Семантические свойства в изоляции показывают менее высокие результаты, однако даже в этом случае качество работы системы превосходит базовый классификатор, выделяющий класс большинства. Интерес вызывает свойство "кластер", которое в нашем случае не оказывает почти никакого положительного эффекта на классификацию.

3. ILP оказывает незначительный положительный эффект на классификацию, и эффект оптимизации наблюдается наиболее отчётливо в случаях, когда исходное качество было невелико.

4. Ограничение на частоту конструкции и увеличение объёма тестовых данных ожидаемо приводят к повышению качества работы системы.

Первые два наблюдения представляют особый интерес, и ниже мы остановимся на них подробнее.

Результаты оценки наборов свойств показывают, что наилучшие комбинации свойств в той или иной форме включают в себя синтаксическое свойство "путь". Это обстоятельство имеет как положительные, так и отрицательные стороны. С одной стороны, мы видим, что свойство "путь"

обладает хорошей предсказательной способностью. Это происходит не в последнюю очередь благодаря синтаксическому формализму, использованному в корпусе СинТагРус, который включает в себя специальные отношения для связей предикатов с их *синтаксическими* актантами. Соответствие между семантическими и синтаксическими актантами не всегда однозначно. Так, в следующем примере акант “Велосипедист” не является синтаксическим субъектом глагола “соблюдать” в поверхностном дереве зависимостей, однако является его семантическим актантом.

Велосипедист *должен быть осторожным, внимательным и строго* ***соблюдать*** *все* ***правила*** *уличного движения .*

Пример 13: Различие между семантическими и синтаксическими актантами

В то же время совпадение синтаксического отношения может служить надёжным индикатором для соответствующего отношения семантического.

Следует отметить, что поскольку обучение и тестирование системы производится на основе автоматического синтаксического анализа, подобная "точность" свойства может оказывать и отрицательный эффект на классификацию в случаях ошибок парсера. Свойство "путь" приобретает высокий вес, и несоответствие пути в тестовом предложении может стать слишком весомым аргументом в пользу неприсвоения роли выбранному актанту. Это обстоятельство, как мы считаем, частично ответственно за приоритет точности над полнотой в нашей системе: в случаях, когда путь от предиката к актанту совпадает, это является очень существенным доводом в пользу присвоения актанту соответствующей роли. В противном же случае роль может быть присвоена на основании других семантических и синтаксических

свойств, однако вес этих свойств относительно невелик, и в большинстве случаев классификатор минимизирует риск и присваивает потенциальному актанту роль "None". Свойство "путь" в нашей системе берёт на себя очень большую смысловую нагрузку, т.к. кроме непосредственно синтаксических отношений регистрирует также дистантные отношения, возникающие, например, в случаях, когда носитель роли находится в соседней клаузе. Предположим, что в тренировочной выборке содержится следующее предложение:

Иван *хочет купить автомобиль, чтобы* ***ездить*** *на нём в деревню.*

Пример 14: Проблема разреженности пути

Здесь путь от целевого предиката до актанта проходит через другой предикат (“*хотеть*” и “*купить*”), и в результате система обучается реагировать на данное значение свойства "путь" приписанием соответствующего класса актанту. При этом в другом случае, аналогичном с точки зрения кореферентности именных групп, система распознает уже другой путь:

Иван *был бы рад купить автомобиль, чтобы* ***ездить*** *на нём в деревню.*

Пример 15: Проблема разреженности пути

Таким образом, если представить, что наш обучающий корпус состоял только из предложений первого типа, а наши тестовые данные содержат предложения второго типа, эти вторые предложения не могут быть проанализированы правильно на основании синтаксиса, т.к. система не может "абстрагировать" пути до уровня, на котором эти пути были бы эквивалентны. Данная проблема может быть решена путём включения в систему компонента

анализа кореферентности или же более гибкого моделирования пути. В нашем случае мы предприняли попытку "генерализации" пути методом усечения путей до средней длины по корпусу. Таким образом мы лишаем систему возможности использовать длинные и нестандартные пути на этапе обучения и стимулируем использование других свойств, например, семантики предполагаемого актанта или его падежа. Как показывают наши результаты, такая стратегия является успешной, и три лучших системы на полных наборах свойств, а также две лучших системы на синтаксических свойствах используют именно короткий путь вместо полного. Мы считаем, что более эффективная генерализация пути, например, на основе вероятностного моделирования последовательностей связей, могли бы помочь улучшить этот результат и сделать свойство "путь" менее разреженным и более содержательным.

Следует отдельно отметить проблему, возникающую при наличии в обучающих и тестовых выборках предложений с эллипсисом, см. например, Пример 16: Имплицитное заполнение роли6:

*Ø Плутала улицами , переулками , **выбежала** к трамваю номер восемь.*

Пример 16: Имплицитное заполнение роли

В этом случае свойство "путь" может быть правильно определено только если синтаксический анализатор поддерживает нулевые элементы, что, насколько нам известно, встречается достаточно редко по причине высокой вычислительной сложности моделей, которые допускают существование нулевых элементов.

На фоне успешной работы конфигураций, основанных на синтаксических свойствах, полностью семантические конфигурации демонстрируют значительно худшее качество работы. По-видимому, это связано в первую очередь с тем, что в отсутствие информации о синтаксисе оказывается

сложным провести границу между представителями классов. Классы актантов оказываются в данном случае практически неотличимы от класса отсутсвия роли с точки зрения свойств, и у классификатора нет возможности провести разграничение между классами на основе только лишь лексической и частеречной информации. В то же время, используя отличные от пути синтаксические свойства, можно обучить систему, всё ещё превосходящую по качеству классификатор на основе большинства В Таблица 14: Лучшие конфигурации без использования свойства "путь представлены пять лучших конфигураций по F1-мере, в которых не используется свойство "путь" на основании данных первого этапа оценки:

Features	P	R	F	Acc
Vform,POS,finncase,lemma,cluster-nouns,case	0.574	0.476	0.491	0.925
Vform,POS,finncase,lemma,cluster-all,case	0.574	0.476	0.491	0.925
Vform,POS,finncase,lemma,prep_lemma,cluster-nouns,case	0.572	0.475	0.490	0.925
Vform,POS,finncase,lemma,prep_lemma,cluster-all,case	0.572	0.475	0.490	0.925
Voice,vform,POS,finncase,lemma,prep_lemma,cluster-all,case	0.568	0.475	0.489	0.924
baseline	0.331	0.356	0.343	0.928

Таблица 14: Лучшие конфигурации без использования свойства "путь"

Как показывает Таблица 14, включение в данные дополнительной информации о падежном оформлении и морфологических характеристиках глагола позволяет эффективно использовать семантические свойства.

Отдельного упоминания заслуживает свойство "кластер", которое, вопреки нашим ожиданиям, по результатам экспериментов не оказывает сколько-нибудь значительного влияния на качество классификации (и даже демонстрирует небольшую отрицательную корреляцию со значениями F-меры). Напомним, что кластер слова в нашем случае определяется с помощью

модели на основе алгоритма Chinese Whispers, построенной на данных из большого корпуса русскоязычных текстов. Существует по крайней мере три проблемы, с которыми мы сталкиваемся при использовании кластеризации в нашей системе.

Во-первых, кластеризация основана на распределении слов из внешнего источника, тексты в котором принадлежат к другой предметной области, нежели материал FrameBank (в основном это новостные тексты, в то время как FrameBank основан на текстах из литературных источников). В результате этого некоторые распределения слов и сходства между словами, на основании которых строится наш граф, могут быть неточны. По этой же причине наша кластеризация не полностью покрывает лексику документов FrameBank. Для кластеризации, включающей все части речи, покрытыми оказывается 70% слов и 42% словоупотреблений, в то время как кластеризация на основе существительных покрывает лишь 20% слов и 37% словоупотреблений. Мы считаем, что это связано с различиями в исходных данных, использованных для построения кластеров и для создания корпуса FrameBank. В случаях, когда кластер для слова не найден, слово не получает метки кластера. Такое отсутствие кластера может оказаться значимым для классификатора: предположим, что все слова-актанты в тренировочном корпусе имели пустую метку кластера, а некоторые слова, которые актантами не являлись, наоборот получили такую метку. В этом случае для классификатора станет значимым *отсутствие* кластера, что может привести к падению качества классификации.

Другая сложность, также связанная с неполным соответствием предметной области, на текстах из которой была построена кластеризация, и предметной области FrameBank – проблема лексической неоднозначности. Поскольку снятие лексической неоднозначности не производится ни в данных FrameBank, ни в данных, на основе которых построена кластеризация, в некоторых ситуациях слову может быть приписан неправильный кластер из-за

несоответствия значений словоупотребления в корпусе FrameBank и тезаурусного входа. Если бы тезаурус и корпус FrameBank принадлежали к одной предметной области, данный эффект мог бы быть частично сглажен действием эффекта "одного значения на дискурс" ("*one sense per discourse*", [Gale, Church, Yarowsky, 1992]), согласно которому в рамках одной предметной области вариативность в значениях слов значительно падает, однако в нашем случае это не так.

Наконец, наша кластеризация страдает от излишней фрагментированности: при высокой точности кластеров, т.е. их внутренней однородности, некоторые очевидно связанные слова оказываются помещены в разные кластеры, как в следующем примере.

Кластер 464	**Кластер 466**
альпинист	журналист
альпинистка	репортер
бейсджампер	коллега
восьмитысячник	телерепортер
...	...

Таблица 15: Проблемы кластеризации

В результате подобной чрезмерной специфичности оказывается невозможным установить связь между словами из первой группы и словами из второй группы на этапе применения системы.

Указанные выше проблемы кластеризации, как нам кажется, могут быть решены путём включения в систему компонента разрешения лексической неоднозначности, извлечения тезауруса из более близкого по содержанию источника, а также доработки процедуры кластеризации. В целом хотелось бы отметить, что требования к качеству лексических свойств оказываются достаточно высокими: для нормального функционирования этих свойств требуется и высокая степень лексического покрытия целевого корпуса, и

совпадения предметной области (что гарантировало бы покрытие в плане значений слов), и высокий уровень генерализации значений. Все эти три задачи находятся за рамками настоящего исследования, однако мы считаем, что их решение представляет несомненный интерес.

Наконец, хотелось бы остановиться на проблемах и возможных усовершенствованиях процедуры ILP-оптимизации, которая обеспечивает приведение вывода системы к формально корректному виду и, как мы могли наблюдать, в некоторых случаях приводит к повышению качества работы системы. Наша формулировка задачи оптимизации включает в себя требования единственности заполнения каждой роли. Исходя из этого требования модуль целочисленного программирования максимизирует суммарную уверенность классификатора по каждому отдельному узлу в дереве зависимостей для выбранного предложения. Отметим, что заполнение каждой роли не требуется, и таким образом в некоторых случаях более надежным решением оказывается приписать слову нулевой класс. В качестве альтернативной стратегии можно было бы рассмотреть введение дополнительного ограничения, которое требует, что каждая роль или определённые роли *были* заполнены. Вместо набора ограничений, который применяется в нашей системе сейчас, а именно

$$\forall j\colon \sum_i x_{ij} = 1$$

$$\forall i \neq \emptyset\colon \sum_j x_{ij} \leq 1$$

Мы бы могли использовать набор следующего вида:

$$\forall j\colon \sum_i x_{ij} = 1$$

$$\forall i \neq \emptyset\colon \sum_j x_{ij} \leq 1$$

$$\forall i \in ObligatoryRoles : \sum_j x_{ij} = 1$$

Однако такой модификации процедуры оптимизации противоречат соображения типологического характера. В русском языке крайне распространено явление эллипсиса, при котором то или иное слово, в том числе актанты, не выражается в поверхностной структуре предложения. Если в английском языке с точки зрения грамматики в таких ситуациях необходимо использовать местоимение, в русском языке может быть использован нулевой элемент, не отражённый в поверхностной структуре.

В этом смысле требование заполнения ядерных ролей хотя и может выполняться формально, на поверхностном уровне не действует. До тех пор, пока нам не будет доступен синтаксический анализатор, который мог бы выделять подобные "нулевые" элементы, требование к заполнению всех ролей для автоматической классификации актантов в русском языке кажется нам слишком строгим. Это утверждение безусловно требует дополнительной проверки, которое находится за рамками настоящей работы.

IV. Выводы

В этой главе мы подведём итоги проведённого исследования, а также опишем перспективные, на наш взгляд, пути дальнейшего развития предложенной системы.

В рамках диссертационного исследования была предложена система автоматической разметки и классификации актантов для русского языка. Это первый опыт построения подобной системы на основе корпуса FrameBank, размеченного по семантическим ролям. В соответствии с установившейся традицией, мы интерпретируем задачу автоматической разметки актантов как задачу классификации и решаем её с помощью методов машинного обучения. В качестве метода классификации мы используем метод опорных векторов (support vector machines) на основе набора лингвистически мотивированных признаков. Для того чтобы получить доступ к лингвистической информации, необходимой для классификации актантов, мы подвергаем тексты исходного обучающего корпуса FrameBank автоматической предобработке: морфологическому анализу, лемматизации и синтаксическому анализу. Отдельный важный этап предобработки – проекция аннотаций FrameBank, выполненных на отрезках текста, на узлы синтаксических деревьев, что позволяет извлекать синтаксические свойства и интерпретировать задачу классификации актантов как задачу классификации узлов дерева зависимостей.

Другой важный этап предварительной обработки данных – фильтрация: поскольку ресурс находится в стадии разработки, некоторые примеры содержат неточности, и мы устраняем подобные примеры с помощью набора фильтрующих правил.

После того как свойства извлечены и выполнена проекция разметки с отрезков текста на узлы дерева зависимостей, мы выполняем классификацию актантов: каждому узлу в дереве зависимостей приписывается семантическая роль или специальная отметка, сигнализирующая о том, что данный узел не несёт никакой роли в выбранном предложении. Недостаток такого подхода состоит в том, что решения классификатора принимаются независимо друг от друга, в результате чего могут нарушаться некоторые теоретические ограничения (например, что одна и та же роль не может быть заполнена дважды для одного предиката). Для решения этой проблемы мы разработали модуль на основе целочисленного программирования, который осуществляет глобальную оптимизацию работы классификаторов с учётом ограничений, налагаемых теорией семантических ролей.

Мы провели детальный анализ работы системы и оценили вклад различных параметров и свойств в качество автоматической разметки актантов. Наш анализ демонстрирует важность синтаксических свойств для автоматической разметки актантов, а также важность соответствия исходной и целевой предметной областей при использовании дистрибутивных моделей для учёта лексического сходства актантов. Наши результаты также демонстрируют, что глобальная оптимизация является важным шагом в автоматической обработке актантов.

Разработка предложенной в работе системы сопровождалась принятием ряда технических решений, обусловленных различными факторами: доступностью и качеством ресурсов, качеством работы модулей предобработки, легкостью анализов результатов работы определенных

подсистем. Принятые решения не являются единственно возможными, и ниже мы остановимся на альтернативных подходах, которые не были использованы в рамках данной работы, но безусловно являются жизнеспособными альтернативыми и представляют интерес для дальнейших исследований по автоматической обработке актантов для русского языка.

Наши выводы о возможных альтернативных подходах к решению автоматической разметки актатнов в русском языке можно условно разделить на три группы. Первая группа выводов связана с решениями, которые находятся в русле используемых в работе подходов, и так или иначе могли бы способствовать развитию и улучшению разработанной системы. Вторая группа выводов касается проблемы использования методов обучения без учителя для решения задач автоматической семантической разметки актантов применительно к русскому материалу. Наконец, финальные замечания связаны с возможными шагами по усовершенствованию корпуса FrameBank. Ниже будет более детально рассмотрена каждая из групп.

IV.1 Альтернативные решения

Интерпретируемый алгоритм машинного обучения

Как уже было сказано, машинное обучение осуществлялось на базе метода опорных векторов. Этот подход обладает рядом безусловных достоинств, наиболее важные из которых – скорость работы и качество получаемых результатов. Основным недостатком метода является сложность интерпретации результатов. При разработке системы в качестве промежуточного этапа мы использовали вариант, в котором вместо метода опорных веторов использовались деревья принятия решений. Результирующее качество классификации этого метода оказалось невысоким, однако благодаря интерпретируемости и прозрачности моделей мы смогли сделать ряд важных

наблюдений, например, о поведении свойства путь и целесообразности усечения пути для повышения обобщающей способности модели. В целом, использование деревьев принятия решений в той или иной форме имеет определённые перспективы применительно к рассматриваемой задаче, особенно в случае, если система будет разрабатываться лингвистами, которые смогут интерпретировать полученные деревья с точки зрения науки о языке. Даже в случаях, когда деревья принятия решений оказываются не самым эффективным методом с точки зрения показателей качества, они могут быть использованы для анализа специфики задачи и адекватности предлагаемых методов. Такой анализ мог бы значительно улучшить понимание проблем автоматической разметки актантов в контексте русского языка.

Усовершенствованная глобальная оптимизация

Модель, использованная нами в рамках данного исследования, достаточно проста с технической точки зрения. Система основана на наборе не зависящих друг от друга классификаторов, результаты работы которых могут быть затем оптимизированы на глобальном уровне с помощью алгоритмов целочисленного программирования. Хотя целочисленное програмимрование и предоставляет определённый уровень глобальной оптимизации, это не единственное возможное решение. Так, в наиболее современных работах по автоматической разметке актантов [Das, 2014] вместо целочисленной оптимизации используются модели, в которых взаимозависимость присвоения ролей включена уже в модель классификации, что представляется нам более естественным и эффективным решением, которое, однако, требует больших усилий по имплементации. Опробирование более современных моделей – один из возможных путей развития предложенной нами системы.

Усовершенствование свойств для обучения

Из уже упомянутых ранее сложностей, с которыми мы столкнулись, хочется ещё раз остановиться на низком вкладе кластеризации и разреженности и специфичности свойства "путь". Отсутствие существенного вклада кластеризации в нашем случае ни в коем случае нельзя считать недостатком выбранной нами семантической модели. Представление RusVectōrēs, на которое мы опирались, позволяет получить высококачественные кластеры, а метод кластеризации на основе алгоритма Chinese Whispers был успешно использован на английском материале. Скорее всего, основная причина низкой эффективности кластерных свойств – высокий вес синтаксических свойств и несоответствие предметных областей корпуса FrameBank и новостных корпусов, на основе которых были созданы модели семантической близости. Проект RusVectōrēs активно развивается, и на сегодняшний день предлагает к использованию модели, построенные на основе Национального корпуса русского языка. Эксперименты с моделями, построенными на различных предметных областях, могли бы увеличить эффективность свойства "кластер" применительно к данным FrameBank. Отметим, что системы, не использующие синтаксической информации, превосходят по качеству базовую систему, которая всегда приписывает актанту класс большинства (т.е. нулевой класс). Мы считаем, что использование лексической информации в автоматической разметке актантов имеет большие перспективы, и полученные нами результаты ни в коем случае не свидетельствуют о том, что семантические свойства слов в русском SRL должны игнорироваться. Другой возможный путь развития – использование готовых семантических ресурсов, например, тезауруса РуТез.

В то же время исследование показало, насколько важно для автоматической разметки актантов свойство синтаксического пути от предиката

к актанту. Отметим, что у подобной опоры на одно свойство могут быть существенные недостатки. Во-первых, свойство "путь" крайне разрежено, что уменьшает способность системы к генерализации: многие пути из тренировочной выборки уникальны, и, будучи ассоциированы с той или иной семантической ролью, могут привести к переобучению классификатора. Во-вторых, это свойство отражает результаты синтаксического анализа, который мог быть проведен с ошибками. Решение, предложенное в рамках данной работы – генерализация пути с помощью усечения при превышении определённого порога длины – является наиболее простым, и даже с его помощью достигается небольше повышение качества работы системы. По-видимому, более сложные механизмы генерализации и фильтрации путей могли бы привести к лучшим результатам. В качестве кандидатов на эту роль кажется разумным рассмотреть замену длинных путей на константное значение, в результате чего для синтаксически удалённых актантов система должна будет опираться на лексические свойства, а также более гибкое усечение путей, например, на основе вероятностных моделей.

Наконец, кажется интересным провести эксперимент по использованию именованных сущностей и словарной семантической информации в качестве свойств для классификации. Семантические роли часто коррелируют с тем или иным семантическим классом объектов, и использование подобных свойств в случаях, когда они доступны, могло бы повысить качество работы системы.

IV.2 Частичное обучение с учителем и обучение без учителя

На сегодняшний день основная часть новых работ по автоматической обработке актантов посвящена использованию методов частичного обучения с учителем и обучению без учителя. Можно выделить две основных причины, по котором данные парадигмы становятся всё более популярными в рамках рассматриваемой области. Методы обучения без учителя позволяют сократить затраты на разработку обучающих ресурсов для SRL. Поскольку корпус FrameBank всё ещё находится в разработке, подобные методы могли бы быть использованы для полуавтоматического расширения корпуса, что значительно сократило бы затраты на разработку и увеличило бы потенциальный объём ресурса. В качестве первого кандидата на использование в этой роли нам видится метод проекции аннотаций, предложенный в работе [Furstenau, Lapata, 2011] и кратко описанный нами ранее. Этот метод позволяет переносить аннотации с предложений, размеченных вручную, на новые предложения, используя синтаксисическое и лексическое сходство слов в качестве опорной информации. Размеченные примеры FrameBank могли бы быть экстраполированы на другие примеры из исходного корпуса, а также на корпуса из других предметных областей, что увеличило бы объём доступных данных и минимизировало бы усилия по поиску новых примеров. В то же время следует отметить, что при использовании этого метода могут возникнуть определенные трудности: во-первых, необходимо разработать модуль разрешения неоднозначности, который автоматически определял бы конструкцию для выбранного предиката. Во-вторых, метод проекции аннотаций позволяет увеличить число примеров, но почти не вносит вклада в разнообразие конструкций, т.к. может только проецировать разметку с конструкций уже существующих. Реализация этого подхода для расширения

корпуса FrameBank – интересная задача для отдельного исследования, и мы надеемся, что подобное исследование в ближайшее время будет проведено.

Существует и другая причина популярности методов обучения без учителя в задачах автоматической разметки актантов. Semantic role labeling – доменно-специфичная задача, и было неоднократно продемонстрировано, что качество работы систем данного класса падает при смене предметной области. Разрабатывать обучающий ресурс для каждой новой предметной области не представляется возможным, и методы на основе частичного обучения и обучения без учителя являются одним из наиболее перспективных направлений в этом контексте. Мы считаем, что на сегодняшний день основной задачей автоматической разметки актантов для русского языка является создание систем на основе обучения с учителем, однако на этапе применения данных систем в реальных приложениях наработки в области методов обучения без учителя также могут оказаться востребованы. Отдельно следует отметить, что обучение без учителя зависит от качества предварительной обработки сильнее, чем обучение с учителем, и в случае использования подобных методов особое внимание, как нам кажется, должно быть уделено качеству синтаксического и морфологического анализа, а также лексико-семантических моделей.

IV.3 Адаптация FrameBank

Наконец, в ходе работы с промежуточной версией корпуса FrameBank мы сделали ряд наблюдений, которые могли бы оказаться полезными как для разработчиков корпуса, так и для будущих исследователей в данной области.

Одна из проблем автоматической разметки актантов, которая редко артикулируется, но в том или ином виде присутствует почти в любом исследовании по этой теме, – это несоответствие целей разработчиков

корпусов и разработчиков систем SRL. Как правило, корпус с разметкой по семантическим ролям создаётся лингвистами и имеет своей основной целью предоставление примеров различных лингвистических явлений, связанных с семантическими ролями. Зачастую корпуса проработаны неравномерно с точки зрения конструкций, а частотное распределение конструкций в корпусах примеров не соответствует действительности. Кроме того, в системах на основе абстрактных ролей (например, FrameNet) существует тенденция к чрезмерному дроблению ролей на основании минимальных лингвистических различий, которые предоставляют теоретический интерес, однако не имеют большого прикладного значения. В то же время, разработчики систем автоматической разметки актантов заинтересованы в том, чтобы роли были однородными и поддавались генерализации, а также в том, чтобы частотное распределение примеров более или менее соответствовало реальному положению дел в выбранной предметной области. Несмотря на то что решить эту проблему полностью не представляется возможным, нам кажется, что сплошная разметка корпуса по семантическим ролям (вместо выбора отдельных примеров) могла бы смягчить оба эффекта: конструкции и роли, которые не встречаются достаточно часто, при таком подходе не размечаются, кроме того, распределение конструкций оказывается более или менее объективным по крайней мере с точки зрения выбранного корпуса. Методы наподобие проекции аннотаций могут оказаться полезными в этом отношении.

Далее, как демонстрирует наша и многие другие работы по автоматической обработке актантов, для выполнения выбранной задачи большую важность имеет синтаксический анализ. Одна из причин популярности систем на основе корпуса PropBank состоит в том, что этот корпус уже содержит синтаксическую разметку. Это позволяет исследователям сконцентрироваться на задаче автоматической разметки актантов и оценивать системы SRL на ручной синтаксической разметке. Такой подход может

показаться искусственным (т.к. при использовании в реальных приложениях, конечно, экспертная синтаксическая разметка доступна не будет), однако кажется правильным предоставить исследователям возможность самостоятельно выбирать между автоматическим и экспертным синтаксисом, а также сравнивать качество SRL в зависимости от этих условий. В этом свете кажется перспективным расширить разметку FrameBank на синтаксический корпус СинТагРус или же добавить синтаксическую разметку в существующие примеры. Проекция аннотаций или аналогичный метод могли бы оказаться полезны и здесь.

Наше исследование показывает, что текущая версия корпуса уже может быть успешно использована для экспериментов по автоматической разметке ролей. В то же время устранение неточностей разметки и публикация корпуса примеров в одном из стандартных форматов (например, CoNLL) оказала бы крайне положительный эффект на развитие этой области для русского языка. Стандартный формат позволил бы применять и сравнивать между собой системы, разработанные для других языков (например, MateTools [Björkelund, Hafdell, Nugues, 2009]), а также упростил бы разработку новых систем. Более того, опора на общепринятый формат позволяет измерять качество работы систем с помощью стандартных оценочных скриптов, применяемых на соревнованиях по автоматической разметке актантов [Hajič и др., 2009; Surdeanu и др., 2008]. Мы считаем стандартизацию критически важным шагом для популяризации выбранной задачи для русского языка, и уверены, что в будущем такой стандартный набор данных будет создан.

Заключение

В рамках диссертационного исследования мы спроектировали и разработали систему автоматической разметки актантов для русского языка. В качестве источника данных был использован корпус примеров FrameNet. Система основана на машинном обучении с учителем и, опираясь на ряд лингвистически мотивированных признаков для описания объектов, выполняет автоматическую классификацию актантов по семантическим ролям. Кроме того, система включает в себя модуль глобальной оптимизации на основе целочисленного программирования,который гарантирует выполнение ряда лингвистических ограничений на приписание семантических ролей. Мы детально проанализировали результаты работы системы и установили вклад различных факторов в качество её работы.

Полученный в результате исследования опыт позволил нам не только определить возможные модификации нашей системы, но и описать в общих чертах пути развития для направления в целом. Нам кажется, что в контексте русского языка наиболее приоритетными задачами на текущий момент являются расширение корпуса примеров для обеспечения репрезентативности и стандартизация данных, которая сделала бы совместную работу в области более эффективной.

В последние годы российская компьютерная лингвистика по спектру решаемых задач приближается к западным стандартам. Были проведены соревнования систем морфологического анализа, синтаксического парсинга, анализа анафоры и определения семантической близости. Автоматическая разметка актантов – ресурсоёмкая задача, но на сегодняшний день все необходимые для решения этой задачи ресурсы в том или ином виде уже представлены на русском материале. Мы надеемся, что с развитием систем предобработки и корпуса примеров FrameBank автоматическая разметка актантов также займёт своё место в стеке технологий, доступных для русского языка.

Библиография

1. Rehurek R., Sojka P. Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks — Valletta, Malta: ELRA, 2010. — С. 45–50.

2. Anisimovich K. V, Druzhkin K.J., Minlos F.R., и др. Syntactic and semantic parser based on ABBYY Compreno linguistic technologies // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 30 мая - 3 июня 2012г.). — Москва: РГГУ, 2012. — С. 810–822.

3. Baker C.F., Fillmore C.J., Lowe J.B. The Berkeley FrameNet Project // Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics — 1998. — Т. 1 — С. 86–90.

4. Ballesteros M., Nivre J. MaltOptimizer: A System for MaltParser Optimization // Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC ’12) — Istanbul, Turkey: European Language Resources Association (ELRA), 2012. — С. 23–27.

5. Bauer D., Fürstenau H., Rambow O. The Dependency-Parsed FrameNet Corpus // Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012) — Istanbul, Turkey: European Language

Resources Association (ELRA), 2012. — С. 3861–3867.

6. Biemann C. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems // Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing — Stroudsburg, PA, USA: Association for Computational Linguistics, 2006a. — С. 73–80.

7. Biemann C. Chinese Whispers Tool [Электронный ресурс]. URL: http://wortschatz.informatik.uni-leipzig.de/~cbiemann/software/CW.html.

8. Björkelund A., Hafdell L., Nugues P. Multilingual Semantic Role Labeling // Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task — Stroudsburg, PA, USA: Association for Computational Linguistics, 2009. — С. 43–48.

9. Blei D.M., Ng A.Y., Jordan M.I. Latent Dirichlet Allocation // Journal of Machine Learning Research — 2012. — Т. 3 — № 4-5 — С. 993–1022.

10. Carnie A. Syntax: A Generative Introduction — Malden, MA: Blackwell Publishing, 2007.

11. Carreras X., Marquez L. Introduction to the CoNLL-2005 Shared Task : Semantic Role Labeling // CONLL 2005: Proceedings of the Ninth Conference on Computational Natural Language Learning — Ann Arbor, Michigan, USA: Association for Computational Linguistics, 2005. — С. 152–164.

12. Castilho R.E. de, Gurevych I. A broad-coverage collection of portable NLP components for building shareable analysis pipelines // Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014 / под ред. N. Ide, J. Grivolla. — Dublin, Ireland: Association for Computational Linguistics and Dublin City University, 2014. — С. 1–11.

13. Chomsky N. Aspects of the Theory of Syntax — Cambridge: The MIT Press, 1965.

14. Chomsky N. Some concepts and consequences of the theory of government and binding — Cambridge: MIT Press, 1982.

15. Christensen J., Soderland S., Etzioni O. Semantic Role Labeling for Open Information Extraction // Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading — Los Angeles, CA, USA: Association for Computational Linguistics, 2010. — С. 52–60.

16. Cortes C., Vapnik V. Support-Vector Networks // Machine Learning — 1995. — Т. 20 — № 3 — С. 273–297.

17. Das D. Frame-semantic parsing // Dissertation Abstracts International, B: Sciences and Engineering. — 2010. — Т. 70. — № 8. — С. 4943.

18. Das D. Statistical Models for Frame-Semantic Parsing // Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014) — Baltimore, Maryland, USA: Association for Computational Linguistics, 2014. — С. 26–29.

19. Das D., Schneider N., Desai C., и др. SEMAFOR 1.0: A probabilistic frame-semantic parser // Technical Report CMU-LTI-10-001. — Pittsburgh, PA, USA, 2010. — С. 1–20.

20. Dowty D. Thematic Proto-Roles and Argument Selection // Language — 1991. — Т. 67 — № 3 — С. 547–619.

21. Fellbaum C. WordNet: An Electronic Lexical Database — London: The MIT Press, 1998.

22. Fillmore C.J. The Case for Case // Universals in Linguistic Theory / под ред. E. Bach, R.T. Harms. — New York: Holt, Rinehart and Winston, 1968. — С. 0–88.

23. Fillmore C.J. Frame semantics // Linguistics in the Morning Calm — Seoul, South Korea: Hanshin Publishing Co., 1982. — С. 111–137.

24. Furstenau H., Lapata M. Semi-supervised semantic role labeling via structural alignment // Computational Linguistics — 2011. — Т. 38 — № 1 — С. 135–171.

25. Gabrilovich E., Markovitch S. Computing semantic relatedness using wikipedia-based explicit semantic analysis // IJCAI International Joint Conference on

Artificial Intelligence — 2007. — Т. 0 — № 0 — С. 1606–1611.

26. Gale W., Church K., Yarowsky D. One sense per discourse // Proceedings of the Workshop on Speech and Natural Language HLT'91 — New York, NY, USA: Association for Computational Linguistics, 1992. — С. 233–237.

27. Gildea D., Jurafsky D. Automatic labeling of semantic roles // Proceedings of the 38th Annual Meeting on Association for Computational Linguistics - ACL '00 — 2000. — № 1972 — С. 512–520.

28. Gildea D., Palmer M. The necessity of parsing for predicate argument recognition // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics — 2002. — № July — С. 239–246.

29. Goldberg A.E. Constructions: a construction grammar approach to argument structure — Chicago: University of Chicago Press, 1995.

30. Gruber J. Studies in lexical relations — Cambridge, MA: MIT, 1965.

31. Haghighi A., Toutanova K., Manning C.D. A Joint Model for Semantic Role Labeling // Computational Linguistics — 2008. — Т. 34 — № 2 — С. 173–176.

32. Hajič J., Ciaramita M., Johansson R., и др. The CoNLL-2009 shared task: syntactic and semantic dependencies in multiple languages // CoNLL '09 Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task — Stroudsburg, PA, USA: Association for Computational Linguistics, 2009. — С. 1–18.

33. Harris Z. Distributional structure // Word — 1954. — Т. 10 — № 23 — С. 146–162.

34. Hirst G. Semantic interpretation and ambiguity // Artificial Intelligence — 1988. — Т. 34 — № 2 — С. 131–177.

35. Jackendoff R.S. Semantics And Cognition — Cambridge: MIT Press, 1983.

36. Johansson R., Nugues P. LTH: semantic structure extraction using nonprojective dependency trees // SemEval'07: Proceedings of the 4th International Workshop on Semantic Evaluations — Stroudsburg, PA, USA: Association for

Computational Linguistics, 2007. — С. 227–230.

37. Johansson R., Nugues P. Dependency-based semantic role labeling of PropBank // EMNLP ’08: Proceedings of the Conference on Empirical Methods in Natural Language Processing — Stroudsburg, PA, USA: Association for Computational Linguistics, 2008. — С. 69–78.

38. Jongejan B., Dalianis H. Automatic training of lemmatization rules that handle morphological changes in pre- , in- and suffixes alike // ACL-2009, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing — Association for Computational Linguistics, 2009. — С. 145–153.

39. Koomen P., Punyakanok V., Roth D., и др. Generalized inference with multiple semantic role labeling systems // CONLL 2005: Proceedings of the Ninth Conference on Computational Natural Language Learning — Association for Computational Linguistics, 2005. — С. 181–184.

40. Kutuzov A., Andreev I. Texts in, Meaning Out: Neural Language Models in Semantic Similarity Tasks for Russian // Компьютерная Лингвистика И Интеллектуальные Технологии: По Материалам Ежегодной Международной Конференции «Диалог» (Москва, 27 — 30 Мая 2015 г.) — Москва: РГГУ, 2015.

41. Land A.H., Doig A.G. An Automatic Method of Solving Discrete Programming Problems // Econometrica — 1960. — Т. 28 — № 3 — С. 497–520.

42. Lang J., Lapata M. Unsupervised semantic role induction with graph partitioning // Proceedings of the Conference on Empirical Methods in Natural Language Processing — 2011. — С. 1320–1331.

43. Levy O., Goldberg Y. Dependency-Based Word Embeddings // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, {ACL} 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers — Association for Computational Linguistics, 2014. — С. 302–308.

44. Levy O., Remus S., Biemann C., и др. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? // Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies — Association for Computational Linguistics, 2015.

45. Lin D. Automatic Retrieval and Clustering of Similar Words // Proceedings of the 17th International Conference on Computational Linguistics - Volume 2 — Stroudsburg, PA, USA: Association for Computational Linguistics, 1998. — С. 768–774.

46. Litkowski K. Senseval-3 task: Automatic labeling of semantic roles // Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text — 2004. — С. 9–12.

47. Liu D., Gildea D. Semantic role features for machine translation // Coling-2010: Proceedings of the 23rd International Conference on Computational Linguistics — Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. — С. 716–724.

48. Lluís X., Carreras X., Màrquez L. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies // Transactions of the Association Computational Linguistics, 1 — 2013. — Т. 1 — С. 219–230.

49. Lluís X., Màrquez L. A joint model for parsing syntactic and semantic dependencies // Proceedings of the Twelfth Conference on Computational Natural Language Learning, CoNLL 2008, Manchester, UK, August 16-17, 2008 — Association for Computational Linguistics, 2008. — С. 188–192.

50. Loukachevitch N. V., Dobrov B. V., Chetviorkin I.I. RuThes-lite, a publicly available version of thesaurus of Russian language RuThes // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 4 — 8 июня 2014 г.). — 2014. — Т. 13 — № 20 — С. 340–349.

51. Lyashevskaya O., Kashkin E. FrameBank: A Database of Russian Lexical Constructions // Analysis of Images, Social Networks and Texts. 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers — Springer International Publishing, 2015. — С. 350–360.

52. MacQueen J. Some methods for classification and analysis of multivariate observations // Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics — Berkeley, Calif.: University of California Press, 1967. — С. 281–297.

53. Malchukov A., Spencer A. The Oxford Handbook of Case — Oxford: Oxford University Press, 2012.— С. 1-960.

54. Marcus M.P., Santorini B., Marcinkiewicz M.A. Building a Large Annotated Corpus of English: The Penn Treebank // Computational Linguistics — 1993. — Т. 19 — № 2 — С. 313–330.

55. Marneffe M.-C. De, MacCartney B., Manning C.D. Generating typed dependency parses from phrase structure parses // Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006) — Association for Computational Linguistics, 2006. — С. 449–454.

56. Màrquez L., Comas P., Giménez J., и др. Semantic Role Labeling as Sequential Tagging // Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005) — Association for Computational Linguistics, 2005. — С. 193–196.

57. McDonald R., Lerman K., Pereira F. Multilingual dependency analysis with a two-stage discriminative parser // Proceedings of the Tenth Conference on Computational Natural Language Learning — Association for Computational Linguistics, 2006. — С. 216–220.

58. McKinney W. pandas: a Foundational Python Library for Data Analysis and Statistics // Python for High Performance and Scientific Computing — , 2011. — С. 1–9.

59. Mel’čuk I. Dependency Syntax: Theory and Practice — New York, NY, USA: State University of New York Press, 1988.

60. Meyers A. Annotation Guidelines for NomBank – Noun Argument Structure for PropBank 2007 — New York, NY, USA: New York University Press, 2007.

61. Mikolov T., Sutskever I., Chen K., и др. Distributed Representations of Words and Phrases and their Compositionality // Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. — , 2013. — С. 3111–3119.

62. Misra Vidyaniwas. The descriptive technique of Panini : an introduction — The Hague: Mouton, 1966.— С. 175.

63. Mitchell S., Sullivan M.O.’, Dunning I. PuLP: A Linear Programming Toolkit for Python — Auckland, New Zealand: University of Auckland, 2011.— С. 12.

64. Ngai G., Wu D., Carpuat M., и др. Semantic Role Labeling with Boosting, SVMs, Maximum Entropy, SNOW, and Decision Lists // Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text — 2004. — № July — С. 183–186.

65. Nivre J., Hall J., Nilsson J. MaltParser: A data-driven parser-generator for dependency parsing // Proceedings of LREC — Association for Computational Linguistics, 2006. — С. 2216–2219.

66. Palmer M., Gildea D., Kingsbury P. The Proposition Bank: An Annotated Corpus of Semantic Roles // Computational Linguistics — 2005. — Т. 31 — № 1 — С. 71–106.

67. Panchenko A., Loukachevitch N. V., Ustalov D., и др. Russe: the First Workshop on Russian Semantic Similarity // Компьютерная Лингвистика И Интеллектуальные Технологии: По Материалам Ежегодной Международной Конференции «Диалог» — М.: РГГУ, 2015.

68. Pedregosa F., Varoquaux G., Gramfort A., и др. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research — 2011. — Т. 12 — С. 2825–2830.

69. Pradhan S., Hacioglu K., Ward W., и др. Semantic role chunking combining complementary syntactic views // Proceedings of the Ninth Conference on Computational Natural Language Learning - CONLL '05 — Morristown, NJ, USA: Association for Computational Linguistics, 2005. — С. 217.

70. Ramshaw L.A., Marcus M.P. Text Chunking using Transformation-Based Learning // Proceedings of the 3rd ACL Workshop on Very Large Corpora — Cambridge MA, USA: Association for Computational Linguistics, 1995. — С. 82–94.

71. Reisinger D., Rawlins K., Durme B. Van. Semantic Proto-Roles // Transactions of the Association for Computational Linguistics — 2015. — Т. 3 — С. 475–488.

72. Roth D. Learning to Resolve Natural Language Ambiguities: A Unified Approach // Proceedings of the National Conference on Artificial Intelligence — , 1998. — С. 806–813.

73. Samuelsson Y., Täckström O., Velupillai S., и др. Mixing and Blending Syntactic and Semantic Dependencies // CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning — Manchester, England: Coling 2008 Organizing Committee, 2008. — С. 248–252.

74. Schapire R.E. A brief introduction to boosting // IJCAI International Joint Conference on Artificial Intelligence — 1999. — Т. 2 — № 5 — С. 1401–1406.

75. Schmid H. Probabilistic Part-of-Speech Tagging Using Decision Trees // Proceedings of the International Conference on New Methods in Language Processing — Association for Computational Linguistics, 1994. — С. 44–49.

76. Schuler K.K. VerbNet: a broad-coverage, comprehensive verb lexicon — Philadelphia, PA, USA: University of Pennsylvania, 2005.

77. Sharoff S., Kopotev M., Erjavec T., и др. Designing and Evaluating a

Russian Tagset // Proceedings of the Sixth Language Resources and Evaluation Conference, LREC 2008 — Marrakech: European Language Resources Association (ELRA), 2008.

78. Sharoff S., Nivre J. The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge // Proc. Dialogue, Russian International Conference on Computational Linguistics — М.: РГГУ, 2011. — С. 591–604.

79. Shen D., Lapata M. Using Semantic Roles to Improve Question Answering // Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) — Association for Computational Linguistics, 2007. — С. 12–21.

80. Sibson R. SLINK: an optimally efficient algorithm for the single-link cluster method // The Computer Journal. — 1973. — Т. 16. — № 1. — С. 30–34.

81. Surdeanu M., Johansson R., Meyers A., и др. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies // Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL ’08) — 2008. — № August — С. 159–177.

82. Surdeanu M., Turmo J. Semantic role labeling using complete syntactic analysis // Proceedings of the Ninth Conference on Computational Natural Language Learning — Stroudsburg, PA, USA: Association for Computational Linguistics, 2005. — С. 221–224.

83. Titov I., Klementiev A. A Bayesian approach to unsupervised semantic role induction // Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics — Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. — С. 12–22.

84. Toldova S., Roytberg A., Ladygina A.A., и др. RU-EVAL-2014 : Evaluating Anaphora and Coreference Resolution for Russian // 20-я Международная

конференция по компьютерной лингвистике «Диалог» — 2014. — С. 1–14.

85. Valin R. Van. Generalized semantic roles and the syntax-semantics interface // Empirical issues in formal syntax and semantics / под ред. F. Corblin, C. Dobrovie-Sorin, J.-M. Marandin. — The Hague: Thesus, 1999. — С. 373–389.

86. Апресян Ю.Д. Лексическая семантика: Синонимические средства языка — Москва: Наука, 1974.

87. Апресян Ю.Д. Типы соответствия семантических и синтаксических актантов // Проблемы типологии и общей лингвистики — СПб, 2006. — С. 15–27.

88. Апресян Ю.Д., Богуславский И.М., Иомдин Б.Л. Синтаксически и семантически аннотированный корпус русского языка: современное состояние и перспективы // Национальный корпус русского языка: 2003—2005 — Москва: Индрик, 2005. — С. 193–214.

89. Апресян Ю.Д., Богуславский И.М., Иомдин Л.Л., и др. Теоретические проблемы русского синтаксиса: Взаимодействие грамматики и словаря — Москва: Языки славянских культур, 2010.

90. Ермаков А.Е., Плешко В.В. Семантическая интерпретация в системах компьютерного анализа текста // Информационные технологии — 2009. — № 6 — С. 2–7.

91. Котельников Д.С., Лукашевич Н.В. Итерационное извлечение шаблонов описания событий по новостным кластерам // Труды 14-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL-2012, Переславль-Залесский, Россия, 15-18 октября 2012 г. — 2012.

92. Ляшевская О.Н., Астафьева И., Гарейшина А., и др. Оценка методов автоматического анализа текста: морфологические парсеры русского языка // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» — 2010. — Т. 9 — № 16 —

С. 318–326.

93. Ляшевская О.Н., Кашкин Е.В. Семантические роли и сеть конструкций в системе FrameBank // Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной конференции Диалог — Москва: РГГУ, 2013. — С. 827–846.

94. Ляшевская О.Н., Кузнецова Ю.Л. Русский фреймнет : к задаче создания корпусного словаря конструкций // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог 2009» (Бекасово, 27-31 мая 2009 г.). — 2009. — Т. 8 — № 15 — С. 306–312.

95. Мельчук И.А. Опыт теории лингвистических моделей “Смысл⇔текст” — М.: Наука, 1974.

96. Мельчук И.А., Жолковский А.К. Толково-комбинаторный словарь современного русского языка. Опыты семантико-синтаксического описания русской лексики — Вена: Wiener Slavistischer Almanach, 1984.

97. Мисюрев А.В., Antonova A.A. Анализатор русскогоязыка Syntautom для соревнования синтаксических парсеров // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 30 мая - 3 июня 2012г.). — Москва: РГГУ, 2012. — С. 823–829.

98. Осипов Г.С., Смирнов И.В., Тихомиров И.А. Реляционно - ситуационный метод поиска и анализа текстов и его приложения // Журнал “Искусственный интеллект и принятие решений” — 2008. — Т. 2 — С. 3–10.

99. Рахилина Е.В. Лингвистика конструкций / / под ред. Е.В. Рахилина. — Москва: Азбуковник, 2010.

100. Смирнов И.В., Shelmanov A.O. Methods for Semantic Role Labeling of Russian Texts // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 4

— 8 июня 2014 г.). — Москва: РГГУ, 2014. — С. 607–619.

101. Теньер Л. Основы структурного синтаксиса — Москва, 1988.

102. Толдова С., Соколова Е., Астафьева И., и др. Оценка методов автоматического анализа текста 2011–2012: синтаксические парсеры русского языка // Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной конференции Диалог — Москва: РГГУ, 2012. — С. 77–92.