Институт системного анализа Федерального исследовательского центра
“Информатика и управление” Российской академии наук

На правах рукописи

Малых Валентин Андреевич

Методы сравнения и построения устойчивых к шуму программных систем в задачах обработки текстов

Специальность 05.13.11 —
«Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор, член-корреспондент Российской академии наук
Арлазаров Владимир Львович

Москва — 2019

Оглавление

Стр.

Стр.

Стр.

Стр.

Введение

В последнее время в мире наблюдается быстрый рост накопления знаний, так называемый “информационный взрыв” [1]. Поток генерируемой информации при этом имеет существенно другой характер, нежели наблюдаемый ранее, а именно, большая часть этого потока содержит шумы разного рода. Например, в случае новостных документов, новостные документы от информационных агентств проходят корректуру и содержат в себе минимальное количество опечаток. Но в настоящее время большая часть новостных документов поступает не от информационных агентств, а от обычных людей, что стало возможно с появлением сети интернет. Тексты из интернета часто содержат опечатки, уровень шума в них составляет 10-15%[2]. Другой пример может быть приведен в распознавании документов, идентифицирующих личность. В случае использования специализированного оборудования точность распознавания может стремиться к идеальной, но существует большое количество документов, изображения которых были получены на бытовые фотокамеры в условиях плохого освещения и несоблюдения условий съемки.

Рассматривая подробнее пример с текстами, создаваемыми обычными людьми, опечатки и орфографические ошибки характерных для таких текстов далее будут называться **шумом**. Для определения величины зашумленности слова берется расстояние Левенштейна [3] от шумной словоформы до исходной. Под *исходной словоформой* понимается грамматически и орфографически корректная словоформа в данном контексте. Под шумной словоформой понимается, соответственно, любая отличающаяся от исходной словоформы.[1] Под расстоянием Левенштейна понимается минимальное количество символов такое, которое надо добавить, убрать или заменить в рассматриваемом слове, чтобы получить целевое. Под целевым словом понимается исходная словоформа, под рассматриваемым - шумная словоформа. Шум может быть порожден различными причинами: как ошибками при наборе текста пользователем, так и ошибками самого пользователя.

[1] Стоит отметить, шумная словоформа может быть грамматически верной, например, такая форма склонения для конкретного существительного допустима, но синтаксически требуется иная форма. Пример: “мама мыла красную раме”.

В существующих системах для решения прикладных задач проблема устойчивости к шуму упускается из виду в виду того, что а) существует проблема открытого словаря - слов с опечатками на много порядков больше, чем словарных слов; б) предполагается, что системы проверки орфографии убирают опечатки из текста. Существующие системы для решения прикладных задач зачастую опираются на векторные представления слов. Существующие распространенные системы векторного представления слов обладают либо полностью ограниченным (закрытым) словарем, как система Word2Vec, либо частично открытым, как система FastText. В работах автора представлены методы построения систем для решения прикладных задач, обладающие открытым словарем и устойчивостью к шуму.

Еще одной проблемой является то, что ранее не было предложено методов сравнения качества систем по их устойчивости к шуму в рассматриваемых в этой диссертационной работе прикладных задачах. Существующие аналоги применяются для систем проверки орфографии и не рассчитаны на варьирование уровня шума, что не позволяет оценить устойчивость систем в разных условиях. В работах автора предложен метод сравнения систем в условиях разного уровня шума применительно к различным прикладным задачам.

Для решения проблемы шумности текстов существуют системы проверки орфографии, широко использующиеся в настоящее время. Но современные системы коррекции орфографии все еще могут ошибаться во многих случаях. Например, для русского языка точность современных систем проверки орфографии в терминах F-меры составляет ниже 85%[4]. Ошибки, допущенные в словах, приводят к ухудшению качества в различных задачах обработки естественного языка. Например, в работе [5] показано, что даже применение промышленных систем проверки орфографии для компенсации шума не дает преимущества перед системой, которая изначально устойчива к шуму. Так как не все опечатки могут быть исправлены или исправлены корректно (как показано, например, в работе Кучержана и Брилла [2] существует некоторое количество исправлений, порядка 1%, некорректных относительно пользовательского намерения, но грамматически правильных), автором разработан альтернативный подход заложения в систему, выполняющую какую-либо задачу устойчивость к шуму, то есть создать систему, не полагающуюся на качество систем коррекции шума.

В настоящей диссертационной работе рассмотрены задачи сравнения качества систем векторных представлений слов, классификации текстов, рас-

познавания именованных сущностей и извлечения аспектов, а также методы построения устойчивых систем для означенных задач.

При разработке программных систем обработки текстов в частности решается задача построения систем *векторных представлений слов*. Системы векторных представлений слов в частности решают задачу моделирования языка. Моделирование языка - это создание модели, которая может предсказать следующее слово, на основании окружающих. Как показано в работе Т. Миколова [6] векторные представления, с помощью которых можно решить эту задачу, также обладают и семантическими свойствами, что делает их полезными в решении других, более высокоуровневых задач обработки естественного языка. Задача *классификации текстов* является классической задачей классификации, где объектом выступает текст, а признаками - входящие в него слова. Задача *распознавания именованных сущностей* - это извлечение или разметка во входящем тексте последовательностей токенов, которые именуют сущности, например, людей или организации.

Извлечение аспектов - это извлечение из входного текста описаний свойств некоторой сущности. Например в предложении “У этого телефона громкий динамик.” сущностью является “этот телефон”, а аспектом - “динамик”. То есть модель извлечения аспектов должна представить на выходе заключение, что в этом предложении содержится аспект “динамик”, так как, как правило, в задаче извлечения аспектов предполагается, что сущность фиксирована.

Актуальность данной работы состоит в том, что несмотря на то, что методы построения устойчивых программных систем предлагались и ранее, но не было разработано методов сравнения качества программных систем для данных задач, который позволяет выбрать лучший метод построения устойчивых к шуму систем.

В настоящее время задачи системы устойчивые к шуму достаточно мало освещены в литературе, но в последнее время появилось две работы посвященные нормализации медицинских концептов (разновидность задачи классификации текстов) [7] и машинному переводу [8].

Объектом данного исследования являются программные системы векторного представления слов, классификаторов текстов, извлечения именованных сущностей и извлечения аспектов, а **предметом** данного исследования является устойчивость к шуму вышеперечисленных программных систем.

Целью данной работы является разработка методов сравнения программных систем по их устойчивости к шуму в разных задачах, а именно в задачах получения векторных представлений слов, классификации текстов, распознавания именованных сущностей и выделения аспектов, а также разработка методов построения программных систем, устойчивых к шуму.

Для достижения поставленных целей необходимо было решить следующие **задачи**:

1. Исследовать устойчивость к шуму существующих программных систем векторных представлений слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов.
2. Разработать методы сравнения программных систем векторных представлений слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов по их устойчивости к шуму.
3. Разработать методы построения программных систем векторных представлений слов, классификации текстов и извлечения аспектов, более устойчивых к шуму, чем существующие аналоги,
4. Реализовать разработанные методы в комплексах программ и получить сравнение устойчивости программных систем к шуму.

Научная новизна:

1. Разработаны новые методы сравнения качества программных систем относительно их устойчивости к шуму для задач векторных представлений слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов. Существующие аналоги разработанных методов применяются для оценки качества систем проверки орфографии и не предназначены для других задач. Также существенным отличием является наличие возможности регулирования уровня шума в разработанных методах.
2. Разработаны новые методы построения программных систем устойчивых к шуму векторных представлений слов, классификации текстов и извлечения аспектов. Разработанные методы применены в описанных задачах и показали во многих экспериментах лучшие результаты.
3. Создан, апробирован и внедрен программный комплекс, реализующий разработанные методы.

Практическая значимость работы заключается в разработанных программных комплексах, реализующих:

– сравнение качества программных систем по устойчивости к шуму;
– построение устойчивых к шуму векторных представлений слов;
– построение устойчивых к шуму методов классификации текстов, распознавания именованных сущностей, извлечения аспектов.

Методы сравнения систем по их устойчивости к шуму могут быть легко адаптированы для широкого круга задач обработки текстов, помимо рассмотренных задач классификации, распознавания именованных сущностей и извлечения аспектов, это могут быть задачи распознавания текстов, распознавания речи, машинного перевода и другие. Разработанный программный комплекс векторного представления слов также может быть применен в широком круге задач обработки текстов.

Методология и методы исследования. Сущность методологии настоящего исследования состоит в формулировании гипотезы о недостаточной устойчивости существующих программных систем для рассматриваемых задач, а также в описании существующих программных и их особенностей, важных для исследуемого аспекта устойчивости к шуму. Описанные особенности программных систем используются в дальнейшем для постановки серий численных экспериментов, что характерно для научного поиска в области информатики в целом. В работе использованы методы теории алгоритмов, теории вероятностей и теории машинного обучения, а именно разделов связанных с теорией нейронных сетей и тематического моделирования.

Основные положения, выносимые на защиту:

1. Разработаны новые методы сравнения качества программных систем относительно их устойчивости к шуму для задач обработки текстов. Шум в виде опечаток встречается во многих существующих текстах. Методы разработаны для задач сравнения программных систем векторного представления слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов на различных языках. Существующие аналоги данного метода не применялись к рассматриваемым задачам.
2. Разработаны новые методы построения устойчивых к шуму программных систем, решающих следующие задачи: построение векторных представлений слов, классификации текстов и извлечения аспектов. Эти задачи часто решаются на текстах, содержащих естественный шум. Разработанный метод в задаче векторного представления слов

позволяет построить системы более устойчивые к шуму на большинстве исследованных приложений векторных представлений, а именно на задачах распознавания парафраз, распознавания логического следования и анализа тональности для русского, английского и турецкого языков. В задаче классификации текстов разработанный метод позволяет создавать программные системы, более устойчивые к шуму, чем существующие аналоги, для русского и английского языков. В задаче извлечения аспектов разработанный метод позволяет создавать системы, более устойчивые к шуму, чем существующие системы на основе нейросетевого и графического подходов к построению таких систем. Разработанные программные комплексы выложены в открытый доступ.

Достоверность Все полученные результаты подтверждаются экспериментами, проведенными в соответствии с общепринятыми стандартами.

Диссертационное исследование соответствует п. 10 “Оценка качества, стандартизация и сопровождение программных систем” паспорта специальности 05.13.11.

Апробация работы. Основные результаты работы докладывались на следующих конференциях:

- 13-я международная конференция о концептуальных решетках и их приложениях (CLA 2016) (18-22 июля 2016 г., г. Москва);
- 5-я международная конференция “Искусственный интеллект и естественный язык” (AINL FRUCT 2016) (10-12 ноября 2016 г., г. Санкт-Петербург);
- 6-я международная конференция по анализу изображений, социальных сетей и текстов (АИСТ 2017), (27-29 июля, г. Москва);
- 13-я международная конференция северо-американского отделения Ассоциации по компьютерной лингвистике (NAACL 2018, без публикации) (1-6 июня 2018 г., г. Новый Орлеан, США);
- 56-я международная конференция Ассоциации по компьютерной лингвистике (ACL 2018) (15-20 июля 2018 г., г. Мельбурн, Австралия).
- Конференция по эмпирическим методам в обработке естественного языка (EMNLP 2018) (31 октября - 4 ноября 2018 г., Брюссель, Бельгия);

– Международная конференция по искусственному интеллекту: приложения и инновации (IC-AIAI-2018) (31 октября - 2 ноября 2018 г., г. Никосия, Кипр);
– Открытая конференция ИСП РАН им. В.П. Иванникова (2018 Ivannikov ISPRAS Open Conference) (22-23 ноября 2018 г., г. Москва)

Публикации. Основные результаты по теме диссертации изложены в 11 печатных изданиях, 7 из которых издано в журналах, входящих в списки ВАК, 6 из которых опубликовано в изданиях, индексируемых Scopus, 4 — в трудах конференций.

Работа [9] опубликована в журнале, включённом в перечень рекомендованных изданий ВАК. Работы [5; 10—14] опубликованы в изданиях, индексируемых в Scopus, при этом работы [5; 14] опубликованы в журнале, включенном в перечень ВАК рецензируемых изданий, входящих в международные реферативные базы. Работа [15] опубликована в издании, индексируемом РИНЦ.

В работе [11] все результаты принадлежат автору. В остальных работах, также все результаты принадлежат автору, однако, в работе [10] Озерину А.В. принадлежат иллюстрации и частично постановка задачи; в работах [9; 12; 14; 15] Лялину В.А. принадлежат описания моделей и часть иллюстраций; в работе [5] Хахулину Т.А. принадлежат описания моделей, Логачевой В.К. вступление и часть иллюстраций.

Личный вклад автора. Все представленные в диссертации результаты получены лично автором.

Объем и структура работы. Диссертация состоит из введения, четырёх глав, заключения, библиографии и двух приложений, словаря терминов и словаря сокращений. Полный объём диссертации составляет 144 страницы, включая 38 рисунков и 11 таблиц. Список литературы содержит 112 наименований.

Содержание работы. Во **введении** обосновывается актуальность исследований, проводимых в рамках данной диссертационной работы, приводится обзор научной литературы по изучаемой проблеме, формулируется цель, ставятся задачи работы, излагается научная новизна и практическая значимость представляемой работы.

Первая глава посвящена обзору работ по методам сравнения систем на предмет устойчивости к шуму, моделированию шума, построению систем, устойчивых к шуму, для различных задач обработки естественного языка. Описаны существующие подходы к моделированию шума для построения моделей.

Описаны общие методы, применяющиеся в обработке естественного языка с акцентом на применении в задачах, где требуется устойчивость к шуму. Существующие методы как правило не предполагают устойчивости к шуму, вместо этого используется проверка орфографии, что в некоторых случаях оказывается недостаточным.

Вторая глава работы посвящена описанию и применению метода сравнения систем по устойчивости к шуму и метода построения устойчивых к шуму систем на примере систем векторных представлений слов. В главе описывается оригинальный метод построения систем устойчивых к шуму на примере системы векторных представлений слов RoVe, генерирующей устойчивые к шуму вектора слов. Данная система сравнивается с широко распространненными системами Word2Vec и fastText, демонстируя их меньшую устойчивость к шуму в различных прикладных задачах на нескольких языках.

В **третьей главе** рассмотрено применение методов сравнения систем по устойчивости к шуму и построения систем устойчивых к шуму в задаче классификации текстов на примере задачи анализа тональности (sentiment analysis). В главе описываются экспериментальные исследования устойчивости к шуму современных систем для классификации текстов, а также предложены расширения одной из популярных систем. Продемонстрировано, что расширения с использованием устойчивых к шуму векторных представлений слов, в частности RoVe, увеличивают общую устойчивость системы для классификации текстов.

Четвертая глава настоящей работы посвящена применению метода сравнения систем по устойчивости к шуму в задаче распознавания именованных сущностей в шумных данных. Для этой задачи на трех языках была протестирована система, показавшая самые высокие результаты на сегодняшний день для английского, русского и французского языков - biLSTM-CRF. Для французского языка описываемая система была применена автором впервые. В работе показано, как разные варианты описываемой архитектуры ведут себя в условиях присутствия шума в тестовых и обучающих данных.

Пятая глава описывает применение методов сравнения систем по устойчивости к шуму и построения систем устойчивых к шуму для проблем выделения аспектов в шумных текстах. В этой главе была исследована лучшая на сегодняшний день модель извлечения аспектов ABAE. В главе описываются расширения предложенной модели и сравнение с базовой моделью LDA.

По результатам экспериментов показано, что расширения с использованием устойчивых к шуму векторных представлений слов, в частности RoVe, более устойчивы к исследуемому шуму.

Шестая глава посвящена теоретическим оценкам алгоритмической сложности для исследуемых моделей. Сделаны выводы относительно зависимости сложности от качества.

Глава 1. Современные методы оценки устойчивости и общие подходы к построению систем обработки текстов

Данная глава посвящена описанию существующих решений для задач построения векторных представлений слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов.

Данная глава содержит в себе 5 подглав. Подглава 1.1 содержит описание моделирования шума. Подглава 1.2 содержит описание существующих методов построения векторных представлений слов, в том числе с учетом шума. Подглава 1.3 содержит описание общих подходов, использованных в работах автора, а именно описание подходов нейронных сетей. Подглава 1.4 содержит описание существующих подходов в области устойчивой к шуму классификации. Подглава 1.5 описывает существующие работы, рассматривающие устойчивость к шуму в задаче распознавания именованных сущностей. Подглава 1.6 содержит описание работ, посвященных извлечению аспектов с учетом влияния шума.

1.1 Моделирование шума и оценка устойчивости к шуму в задачах обработки естественного языка

Моделирование шума используется в различных задачах обработки естественного языка. Основным применением для моделирования шума является задача проверки/исправления орфографии. Так, в работе Карлсона [16] шум определяется, как вставка, удаление и замена букв в слове. Для каждого из типов шума берется фиксированная вероятность 0,05. В случае добавления символа количество символов для вставки сэмплируется из нормального распределения с параметрами $\mathbb{N}(0; 0{,}3)$ с округлением полученного значения вверх о ближайшего целого. Для каждого из этих символов выбирается позиция для вставки перед каким-то символом, включая специальную конечную позицию в слове.

В работе Ислама [17] с вероятностью 0,005 слово заменяется на свою вариацию с расстоянием Левенштейна 1. Дополнительно к исходному определению расстояния Левенштейна авторы используют транспозицию соседних символов.

В этой работе есть важное ограничение, авторы рассматривают только те вариации написаний для слов, которые являются нормативными (возможно, для других слов). В работе [2] модель ошибок берется также, как расстояние Левенштейна равное 3, но без ограничений предыдущей работы.

Работа Ниу [7] из другой области, а именно из области классификации текстов. Модель шума в этой работе состоит из 4 позиций: добавление символа '#' перед словом, удаление случайного символа, удвоение случайного символа, случайная транспозиция одного символа. Эти шумы призваны моделировать шумы в социальных сетях. В качестве модели применения выбрана следующая: из всего документа (сообщения в социальной сети) выбирается одно слово и к нему применяется операция из списка.

В работе Белинкова и Блинка [8] рассматриваются следующие 4 модели ошибок:

- перемена местами двух случайных соседних букв в слове,
- перемешивание всех букв в слове, кроме первой и последней,
- перемешивание всех букв слове, без исключений
- замена одной буквы в слове на соседнюю на клавиатуре.

В работе была продемонстрирована неустойчивость современных систем машинного перевода к такого рода шумам и их комбинациям. Стоит отметить, что разные уровни шума в работе не исследовались. Из опубликованного исходного кода можно заключить, что уровень шума, при котором были произведены эксперименты равен 100% на слово, т.е. шумы обязательно применялись к каждому входному слову, а значит устойчивость к разным уровням шума в этой работе не исследовалась. Также стоит отметить, что предлагаемый в данной работе метод является расширением этого метода, позволяющим не только качественно оценить устойчивость, но и установить какой уровень шума в данных является приемлемым для рассматриваемых моделей.

В работе Джуравски (Jurafsky) и соавторов, посвященной исправлению грамматических ошибок [18], одним из типов используемых шумов является добавление опечаток в слова. Авторы используют удаление и вставку произвольного символа. Уровень шума они фиксируют на основании вычисленного среднего расстояния Левенштейна для используемого параллельного корпуса. Исследований по разным уровням шума в работе не приводится.

В работе Хайралла (Khayrallah) и Кёна (Cohn) [19] исследовались характерные шумы для задачи машинного перевода. Описываемые в работе типы

шумов сводятся к ошибкам разметки или определения языка, за исключением шума в виде замены слова. К сожалению, шум в виде опечаток в этой работе не исследовался. Еще в одной работе, посвященной машинному переводу авторства Хейгольда (Heigold) и соавторов [20] рассматриваются различные виды опечаток, а именно замена букв в слове и два типа перемешивания букв. Описанные шумы применяются для определения устойчивости систем машинного перевода. Важно отметить, что в работе также используется понятия уровня вводимого шума. Данный метод сравнения моделей в целом повторяет предложенный автором в работе [11] в 2016 году.

1.2 Векторные представления текстов и слов

В этом разделе будет описаны классический подход к векторизации текстов TF-IDF, применяющийся до сих пор в качестве базового; подходы к построению векторных представлений Word2Vec и Glove, а также несколько подходов, базирующихся на описанных подходах Word2Vec и Glove.

1.2.1 TF-IDF

Подход TF-IDF (“частота токена - обратная документная частота”) - это классический подход, предложенный в 1988 году [21], тем не менее до сих пор представляющий собой систему, качество которой позволяет ее использовать в экспериментах в качестве базовой. В основе подхода лежит умножение двух членов:

– **TF** (term frequency) или *частота токена* - это частота с которой токен (слово) встречается в конкретном документе; эта частота отражает важность токена для данного конкретного документа;
– **IDF** (inversed document frequency) или “обратная документная частота” - это частота встречаемости токена в документах рассматриваемого корпуса, возведенная в степень -1 и прологарифмированная; эта часто-

та отражает специфичность токена для корпуса (более специфичные токены - более важны).

TF-IDF описывается следующей формулой:

$$TF - IDF(w, d) = \frac{count(w, d)}{\sum_{w' \in V} count(w', d)} \cdot log(\frac{\mathbb{1}_w(d)}{\sum_{d' \in C} \mathbb{1}_w(d')})$$

где w и w' - токены из словаря V, d и d' - документы из корпуса C, $count(w,d)$ - частота токена w в документе d, $\mathbb{1}_w(d)$ - индикаторная функция для присутствия токена w в документе d.

Отдельные документы (или другие последовательности, такие, как предложения) могут быть представлены в векторном виде при помощи TF-IDF, как вектор длины словаря, с ненулевыми значениями (а именно значениями функции TF-IDF) для слов, встретившихся в документе; значения остальных компонентов вектора (соответствующих невстретившимся словам) полагаются в простом случае равными нулю. На практике часто применяется так называемое “сглаживание”, а именно приписывание невстетившимся словам некоторой константы для избежания вычислительных сложностей, связанных с делением на ноль.

1.2.2 Система Word2Vec

Классической уже системой для получения векторных представлений слов является система Word2Vec [23]. В этой работе был предложен подход под названием CBOW (continuous bag of words), изображенный на рис. 1.1. Вместо статистического моделирования языка при помощи n-грамм вводится новая постановка задачи — предсказание слов по их контекстам.

В подходе CBOW задача ставится следующим образом: для каждого слова из контекста берется его векторное представление, эти векторные представления усредняются и получившееся представление используется, как вектор признаков для классификатора по всем словам словаря, который должен предсказать центральное слово по его контексту.

Обучается данная система, например, с помощью стохастического градиентного спуска. Функционал, который оптимизируется в процессе обучения CBOW (логарифм правдоподобия распределения слов над контекстами):

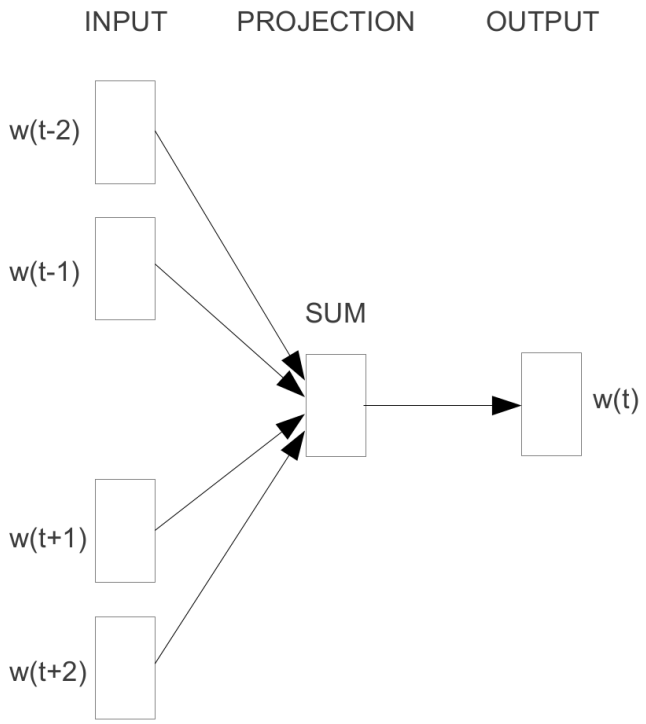

Рисунок 1.1 — Архитектура модели continuous bag of words. Вектора контекстных слов подаются на вход модели, на выходе ожидается предсказание центрального слова [22].

$$L = \frac{1}{N}\sum_{i}^{N} ln(p(w_i|C(w_i))) \to \max \tag{1.1}$$

$$p(w_i|C(w_i)) = \underset{w_i \in W}{softmax}(\sum_{w_k \in C(w_i)} v_{w_k}^{\top} u_{w_i}) \tag{1.2}$$

В этой же статье был предложен еще один вариант построения векторов Skip-Gram рис. 1.2.

В этом подходе авторы предлагают получать дискретное распределение на контексты по данному слову. То есть для входного слова w модель выдает вероятность появления каждого слова в контексте слова w. Функционал, который используется для обучения модели:

$$L(U,V) = \frac{1}{N}\sum_{i}^{N}\sum_{w_k \in C(w_i)} p(w_k|w_i) \to \max_{U,V} \tag{1.3}$$

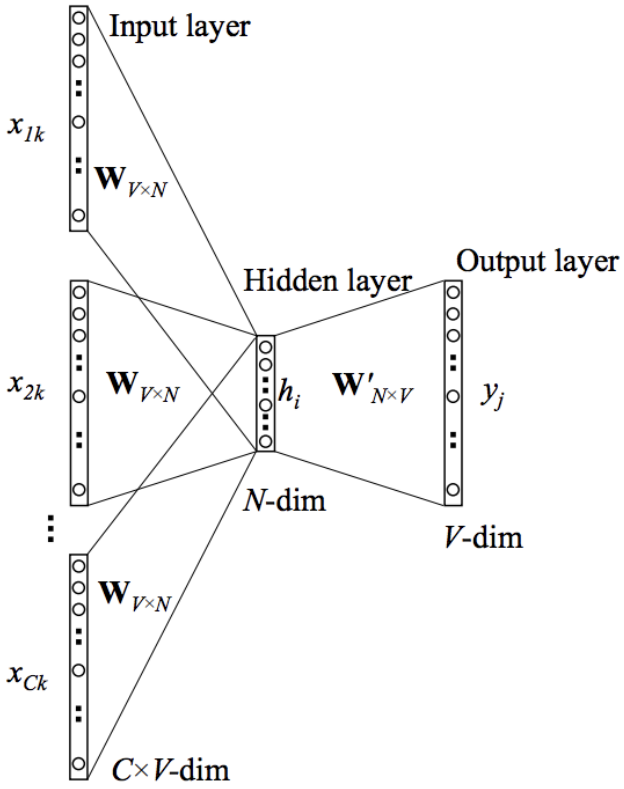

Рисунок 1.2 — Архитектура модели Skip-Gram. В этой модели по центральному слову предсказываются все его контекстные слова. [22]

$$p(w_k|w_i) = \underset{w_k \in W}{softmax}(v_{w_k}^{\top} u_{w_i}) \tag{1.4}$$

Важным эмпирическим свойством системы модели, является семантическая интерпретируемость математических операций над векторами из построенного представления. Пример такой семантической интерпретируемости представлен на рисунке 1.3.

Система Word2Vec имеет важный недостаток, заключающийся в том, что она работает только со словами из словаря. То есть система Word2Vec может выдать векторное представление только для такой словоформы, которая явно была указана в момент построения модели. Вторым важным недостатком системы является то, что эта система не использует локальный контекст для сопоставления вектора слову. Использование же этой информации позволяет повысить качество, как показано в работе Хуанга [24].

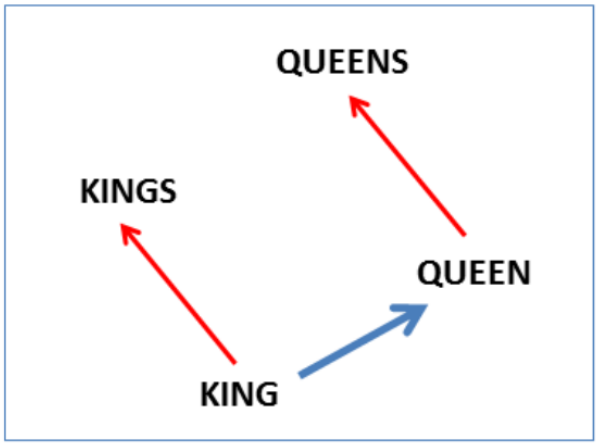

Рисунок 1.3 — Семантические свойства, наблюдаемые в системе word2vec для английского языка [6]. Вектор слова “king” (ед.ч.) относится к вектору слова “kings” (мн.ч.), также, как вектор слова “queen” (ед.ч.) относится к вектору слова “queens” (мн.ч.)

Сэмплирование негативных примеров

Система Word2Vec была улучшена в дальнейшем в работе автора оригинальной системы Т. Миколова [25]. Одним из самых значимых дополнений к системе было сэмплирование негативных примеров (negative sampling).

В стандартной модели CBoW предсказываются и оптимизируются вероятности слов при условии контекста. Функция SoftMax при расчете по всему словарю производит достаточно много лишних вычислений, т.к. существенное количество слов не встречается в контексте друг друга. Был предложен вариант оптимизации вычислений, который получил название сэмплирование негативных примеров (Negative Sampling). Изначально эта идея высказана в работе Смита [26], но в настоящей диссертационной работе будут использоваться обозначения из статьи Т. Миколова [6]. Суть этого подхода заключается в том, что максимизируется вероятность присутствия для целевого слова в типичном контексте и одновременно минимизируется вероятность встречи в нетипичном контексте. Функционал негативного сэмплирования имеет вид:

$$L(x) = log(\sum_{i \in C} e^{-s(x,w_i)}) + log(\sum_{j \notin C} e^{s(x,w_j)}) \tag{1.5}$$

Здесь $s(x,w)$ - это функция схожести для векторов. Схожесть векторов должна соответствовать семантической близости слов в смысле дистрибутивной гипотезы [27]. В дальнейшем в качестве функции схожести будет использоваться косинусная мера. Данную мера описывается следующей формулой:

$$s(x,w) = cos(x,w) = \frac{< x,w >}{\|x\|\|w\|} \tag{1.6}$$

В формуле 1.5 явным образом можно выделить две части: позитивную (где функция схожести берется со знаком '-', т.к. это функция потерь) и негативную.

Позитивная часть отвечает за типичные контексты для рассматриваемого слова, это распределение совместной встречаемости слова и остальных слов корпуса. Негативная часть – это набор слов, которые с нашим целевым словом встречаются редко (либо не встречаются вообще). Этот набор порождается из распределения, которое на практике берется как равномерное по всем словам словаря корпуса. Интересно, что данный подход ускоряет сходимость в несколько раз.

Стоит отметить, что сэмплирование негативных примеров не устраняет вышеуказанные фундаментальные недостатки системы, а именно зависимость от конкретного написания слова и неучет контекста в момент использования системы.

1.2.3 Модель fastText

Группой ученых во главе с Т. Миколовым, которой ранее были написаны работы про систему Word2Vec, был также предложен другой подход – fastText, в работе [28], где вопрос о несловарных словах был разрешен путем создание вектора слова через суммирование векторов для n-грамм, из которых состоит слово. Данный подход включает в себя подход Word2Vec (конкретно модель SkipGram) и использует технику сэмплирования негативных примеров. К сожалению, полностью вопрос OOV в этой системе не разрешается, т.к. в этой модели используется словарь n-грамм символов, который по определению ограничен и существует вероятность, что n-граммы слова там не будут найдены. Помимо

этого, эта система наследует от системы Word2Vec свою фиксированную природу в плане того, что игнорирует существующий вокруг слова контекст.

Авторы предлагают использовать информацию не только о целом слове, но и о его составляющих: вектор для слова в этой модели зависит от входящих в слово триграмм. Способ использования информации о n-граммах полностью аналогичен системе Word2Vec, только в качестве контекстных слов здесь выступают триграммы слова, а в качестве центрального слова - само изначальное слово. В качестве сильной стороны этой системы стоит отметить, что для обучения векторов триграмм количество данных больше на половину порядка (при типичной длине слова 5-6 букв для русского и английского языков [29], такое слово содержит 3-4 триграммы), что дает возможность обучать вектора триграмм даже на сравнительно небольших корпусах. Эта система, несмотря на то, что она ставит своей целью работу с редкими словами, в некоторой степени применима и для работы с опечатками, если слово с опечаткой рассматривать, как редкое.

1.2.4 Система GloVe

Стоит упомянуть еще одну широко используемую систему. Данная архитектура была предложена исследовательской группой в Стенфордском университете под руководством К. Мэннинга [30]. Она позволяет для каждого слова получить вектор фиксированной длины основываясь на статистических данных об этом слове.

Пусть объем словаря равен V. Для всех слов, нумерующихся от 1 до V, составлена матрица слово-слово $X \in \mathbb{R}^{V \times V}$, где x_{ij} - количество раз, которое слово i встретилось в контексте слова j. Размер контекстного окна является гиперпараметром модели.

Обозначим $X_i = \sum_k x_{ik}$. Тогда вероятность того, что слово j встретилось в контексте слова i составляет $p_{ij} = \dfrac{x_{ij}}{X_i}$.

Требуется построить такую функцию $f(w_i, w_j, \hat{w}_k)$, где $w_i, w_j, \hat{w}_k$ векторные представления для слов i, j и k соответственно, чтобы она показывала, какое из слов i или j более вероятно встретится вместе со словом k.

Авторы GloVe предлагают использовать функцию вида:

$$f((w_i - w_j)^\top \hat{w}_k) = \frac{f(w_i^\top \hat{w}_k)}{f(w_j^\top \hat{w}_k)}, \tag{1.7}$$

где $f(w_i^\top \hat{w}_k) = p_{ik}$.

Далее предлагается в качестве $f(\cdot)$ взять $f(w) = exp(w)$, а сам вектор слова подбирать так, чтобы

$$w_i^\top \hat{w}_k = \log p_{ik} = \log x_{ik} - \log X_i$$

учитывая, что $\log X_i$ величина фиксированная и представима в виде $\log X_i = b_i + \hat{b_k}$. Затем авторы предлагают обучать веса w_i не через простую сумму квадратов отклонений, а через взвешенную сумму. Таким образом записывается целевой функционал для оптимизации:

$$J = \sum_{i,j=1}^{V} c(X_{ij})(w_i^\top \hat{w}_k + b_i + \hat{b_j} - \log x_{ij}), \tag{1.8}$$

где функция $c(x_{ij})$ будет просто не присваивать частым совместным встречаемостям слишком больших весов.

В работе [30] показано, что выдаваемые системой GloVe векторные представления показывают себя в задачах нахождения синонимов, гипонимов и гиперонимов лучше или не хуже, чем модель Word2Vec. Также стоит отметить, что для тренировки системы, дающей высокое качество в упомянутых задачах требуется меньший, чем системе Word2Vec, корпус данных. Однако, в этой системе присутствуют те же проблемы, что и в модели Word2vec по обработке слов не из словаря и неучета локального контекста.

1.2.5 Другие системы векторного представления слов

Представленная в работе Линя и соавторов [31] система векторного представления слов обладает открытым словарем - вектора слов вычисляются с помощью RNN над векторными представлениями символов. Авторы утверждают, что их система неявно изучает морфологию, что делает ее пригодной для морфологически богатых языков. В качестве доказательств они приводят результаты экспериментов на задаче моделирования языка и разметки частей

речи в предложении. Однако, никаких экспериментов на устойчивость к опечаткам в работе не представлено - все эксперименты проводились на широко известных “чистых” корпусах (то есть корпусах заведомо не содержащих опечаток), например, на корпусе Penn Tree Bank.

Другой подход заключается в обучении системы, которая аппроксимирует исходные векторные представления слов и кодирует неизвестные ей слова в то же векторное пространство. Ю. Пинтер и соавторы [32] применяют двунаправленные рекуррентные сети (конкретно LSTM) для того, чтобы из векторных представлений символов, входящих в слово получить векторное представление, подобное заранее заданному. Этот подход требует заранее известных векторных представлений, но обеспечивает наличие векторных представлений для OOV слов. Как и в предыдущем подходе, эффективность предложенного подхода продемонстрирована на задаче разметки частей речи на чистом корпусе, обучение проводилось на части корпуса, чтобы искусственно создать ситуацию с несловарными словами.

Р. Астудилло в работе [33] проектирует предварительно обученные векторные представления слов в более низкоразмерное пространство - это позволяет им обучать значимые вектора слов для новых слов из малого объема данных.

Однако начальные вектора слов, необходимые для этих подходов, не могут быть обучены на зашумленных данных без необоснованного расширения словаря редкими вариантами слова с опечаткой. Основная сложность в таком подходе, что для слова типичной длины 5 символов есть существенное количество опечаток - если предположить, что одна из пяти букв может быть заменена на любую другую, то это для латинского алфавита дает $27^5 = 14348907$ вариантов. Даже на таком упрощенном примере становится понятно, что словарь, в который включаются слова с опечатками разрастается. Следующая проблема заключается в том, что слово с опечаткой может встретиться в корпусе всего один раз, что не позволяет выучить для такого слова его векторное представление.

Для борьбы с шумными обучающими данными Нгуен и соавторы в работе [34] обучают нейронную сеть, которая фильтрует векторные представления слов. Это делает их более устойчивыми к статистическим артефактам в обучающих данных. Однако модель все еще имеет фиксированный словарь, вектора слов для статических выборов просто не производятся.

Существует большое количество работ по векторным представлениям, нацеленным на неизвестные системе словоформы. В работе [35] Е. Выломова и соавторы обучают систему построения векторов над словами, которые объединяются в слова через рекуррентную (RNN) или сверточную (CNN) нейронную сеть. Атомарными единицами в данном случае являются символы или морфемы. Морфемы дают лучшие результаты в задаче перевода, в частности для морфологически богатых языков. Этот метод дает вектора высокого качества, но требует для подготовки модели морфологического анализа. Этот подход также не рассчитан на наличие опечаток.

Неизвестные слова обычно представлены подсловами (морфемами или символами). Эта идея широко используется в исследованиях по системам векторных слов. Это не только дает возможность кодировать слова OOV, но также было показано, чтобы улучшить качество векторного представления. В работе Чжана [36] были получены результаты, показывающие, что векторные представления на уровне символов, обученные с помощью CNN, могут хранить информацию о семантических и грамматических особенностях слов. Авторы протестировали эти векторные представления на нескольких последующих задачах. Саксе и соавторы в работе [37] использовали CNN на уровне символов для обнаружения нарушений в слове, а Верман в работе [38] построил независимую от языка систему для анализа тональности с помощью векторных представлений на уровне символов, которые невозможно построить на уровне слов.

Еще до появления системы Word2Vec было высказано предположение о том, что контекст слова имеет значение не только в целом (т. е. контекст слова определяет его значение), но и в каждом случае использования слова, например, в работе Хуанга [24]. Это привело к появлению векторных моделей слов, которые дают векторные представления слов относительно локального контекста. Существует множество доказательств того, что контекстуализация (то есть учет влияния контекста) предварительно подготовленных векторных представлений улучшает их, например, работа Киела [39] и повышает качество последующих задач, например, машинного перевода, как показали Р. Сохер и соавторы в работе [40] или вопросно-ответных систем, как показал Питерс в работе [41].

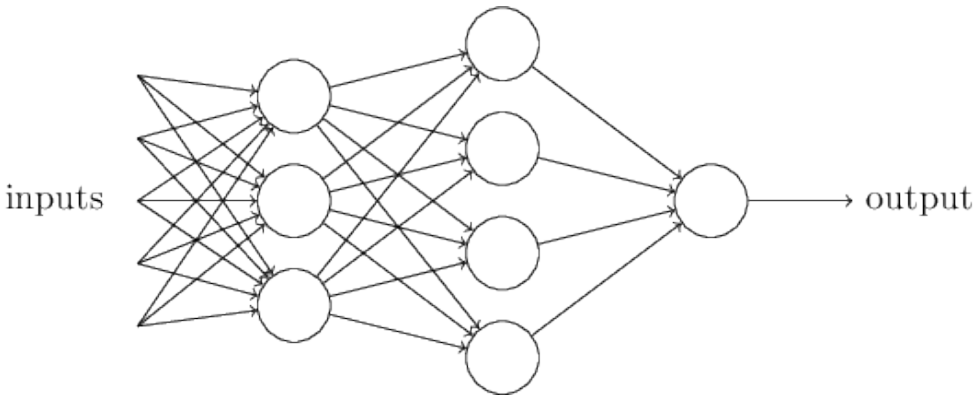

Рисунок 1.4 — Пример архитектуры нейронной сети. Взято из работы [42].

1.3 Общие подходы к построению нейронных сетей

В этом подразделе рассматриваются методы глубокого обучения, то есть обучения нейронных сетей различным задачам машинного обучения. Две главных темы, описанные в этом подразделе - рекуррентные (1.3.2) и сверточные (1.3.3) нейронные сети, представляющие собой два основных существующих подхода в области нейронных сетей на сегодняшний день.

1.3.1 Полносвязные нейронные сети

Самый простой вид нейронных сетей - так называемые полносвязные сети (FCN). FCN - это преобразование входного вектора к выходному путем умножения входного вектора на матрицу весов и применения к результату некоторой нелинейной функции [43; 44], как показано в уравнении 1.9.

$$y = f(W \times x + b), \tag{1.9}$$

где x - входной вектор, y - выходной вектор, W - матрица весов (слоя) нейронной сети, b - вектор весов нейронной сети, $f(\cdot)$ - произвольная нелинейная функция; часто на практике используются сигмоидальная функция и функция гиперболического тангенса.

На рис. 1.4 изображен пример простой полносвязной нейронной сети. Нейронная сеть состоит из входов, которые передают информацию на первый

(входной) слой. Далее следуют один или несколько скрытых слоев. С последнего скрытого слоя информация передается на выходной слой, который непосредственно выдает ответ.

Полносвязные сети практически не применялись в задачах обработки естественного языка до последнего времени [45], но сейчас снова получают распространение [46].

Функции активации

В процессе развития нейронных сетей было предложено множество функций активации, например, softplus, представленная в работе Наира [47], или гиперболический тангенс ЛеКуна, представленный в работе [48]. Но наиболее широко используемыми до сегодняшнего времени являются сигмоидальная функция и гиперболический тангенс, берущие свое начало в математическом анализе. Сигмодиальная функция:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{1.10}$$

Ее особенностью является то, что производная этой функции выражается через нее саму:

$$\sigma'(x) = 1 - \sigma^2(x). \tag{1.11}$$

Аналогично для гиперболического тангенса:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1.12}$$

Его производная выражается аналогично функции $\sigma(x)$:

$$tanh'(x) = 1 - tanh^2(x). \tag{1.13}$$

Процедура принудительного забывания

Дропаут (dropout) - это специальное преобразование накладываемое на входные веса любого слоя нейронной сети, которое препятствует поступлению

информации на вход того слоя, к которому она применена. В этом смысле процедура дропаут является процедурой принудительного забывания. Эта процедура была описана в работе Сривастава и соавторов [49]. Для матрицы весов W размером $k \times l$, где k - количество нейронов в слое, а l - количество входных весов для каждого из нейронов в слое, с помощью испытаний Бернулли генерируется бинарная матрица D аналогичного размера $k \times l$. Матрица W перемножается по Адамару с матрицей D и полученные значения используются вместо матрицы W. Матрица D может генерироваться для каждого объекта из обучающей выборки, но как правило она фиксируется для всех объектов входящих в один мини-батч.

Дропаут используется в процессе обучения нейронной сети для избежания так называемой ко-адаптации нейронов, как показано в работе Н. Сривастава [49]. В процессе тестирования нейронной сети эта процедура не применяется.

1.3.2 Рекуррентные нейронные сети

В то время, как полносвязные нейронные сети являются простейшим и в некотором роде наиболее общим видом нейронных сетей, существуют также более специфичные типы нейронных сетей, полезные в отдельных задачах. Одним из таких специфических типов нейронных сетей являются рекуррентные нейронные сети (RNN), предложенные Ф. Пинеда в работе [50]. RNN применяются для обработки данных, в которых есть выраженная последовательность, таких как временные ряды или последовательности слов. Ключевой особенностью этой архитектуры является обмен информацией между шагами по времени. В используемых в настоящее время вариантах RNN существует понятие состояния RNN, которое переходит на следующий временной шаг с текущего.

$$\mathbf{y}_t = V \times f_H(\mathbf{s}_t), \tag{1.14}$$

где

$$\mathbf{s}_t = \mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{s}_{t-1} + \mathbf{b}_h, \tag{1.15}$$

$f_H(\cdot)$ - функция активации для скрытого слоя нейронной сети, $\mathbf{b}_h$ - вектор смещения для скрытого слоя. Матрицы V, W, U - матрицы весов нейронной сети.

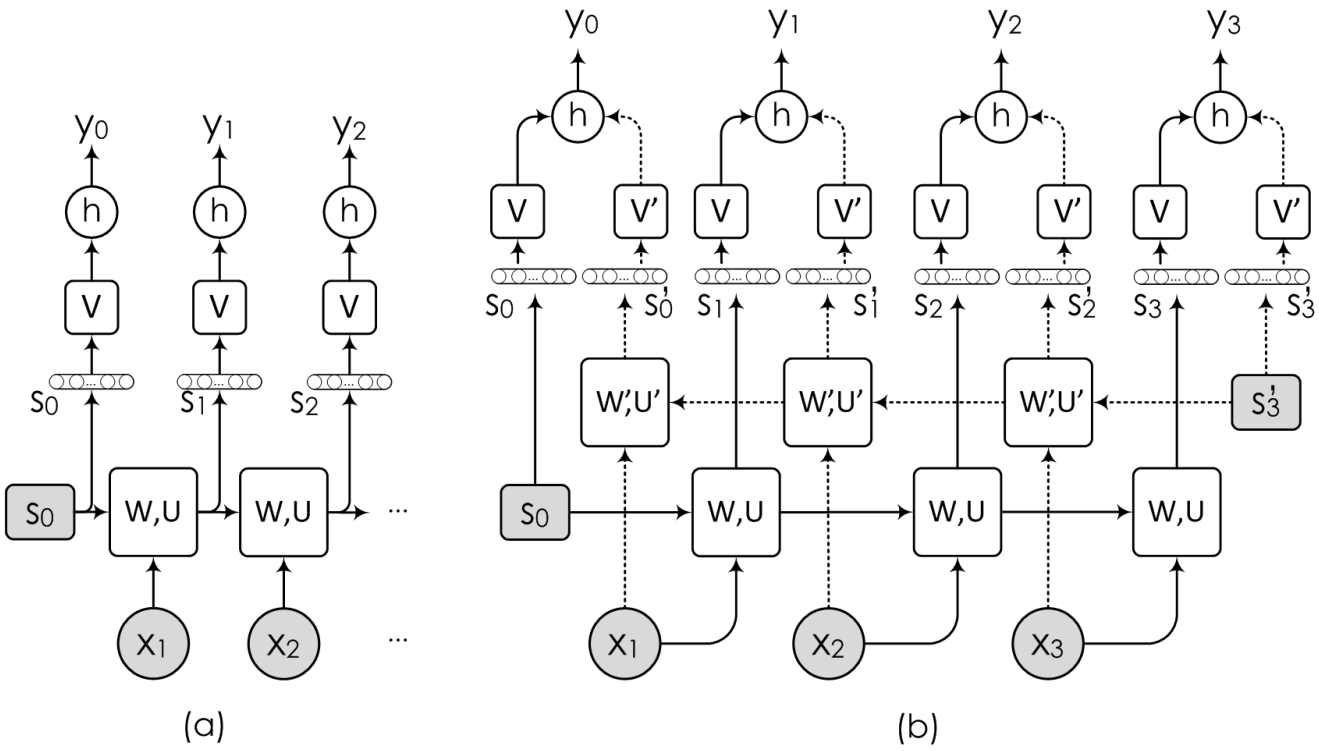

Рисунок 1.5 — Рекуррентные нейронные сети: (a) обычная RNN; (b) двунаправленная RNN.

Описанный механизм представлен на рис. 1.5a. Стоит отметить, что оригинальные рекуррентные сети в настоящее время не применяются на практике в силу того, что демонстрируют заведомо более низкие результаты, как показано в работе К. Чо [51], в силу этого далее в разделе будут описаны применяемые в настоящее время варианты рекуррентных элементов нейронных сетей.

Двунаправленная рекуррентная нейронная сеть

Рекуррентные нейронные сети приспособлены для обработки последовательностей в одном направлении. Но как человек при чтении может возвращаться назад, так и две RNN могут образовывать конструкцию позволяющую уменьшить забывание поданного текста, которое является большой проблемой всех рекуррентных сетей, но в первую очередь классических, как это показано в работе Бенжио [52]. В работе Шустера [53] была предложена модель двунаправленной рекуррентной нейронной сети, обладающей большей обобщающей способностью. Для этого одной из RNN текст подается в обратном направлении.

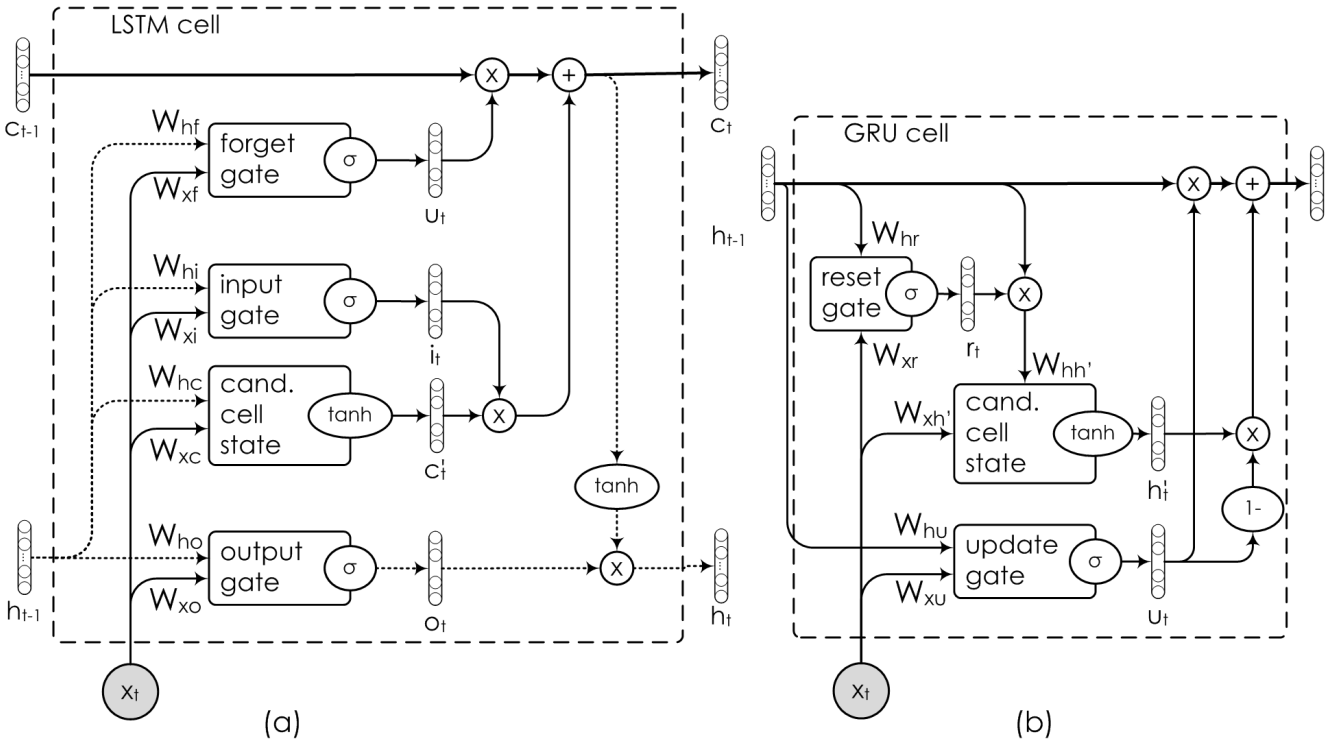

Рисунок 1.6 — Современные архитектуры RNN: (a) LSTM; (b) GRU.

Выходы обеих RNN для соответствующей последовательности конкатинируются вместе. Данный подход изображен на рис. 1.5b.

Долговременная краткосрочная память

Долговременная краткосрочная память (Long Short-Term Memory, LSTM) - это название архитектуры, которая была вдохновлена устройством человеческого мозга, а точнее его краткосрочной памяти, как это описано в работе Хохрайтер и Шмидхубера [54]. Главная идея, лежащая в основе LSTM - это необходимость иметь некую ячейку памяти для хранения информации, а также в механизме работы с этой памятью. Этот механизм состоит из двух, как в работе [54], или трех, как в работе Герса и Шмидхубера [55], нелинейных функций, применяемых к:

- входу, иными словами - что должно добавиться к текущей информации в памяти,
- выходу, т.е. насколько сильным должен быть выходной сигнал,
- и забыванию, т.е. сколько информации будет утеряно в процессе обработки.

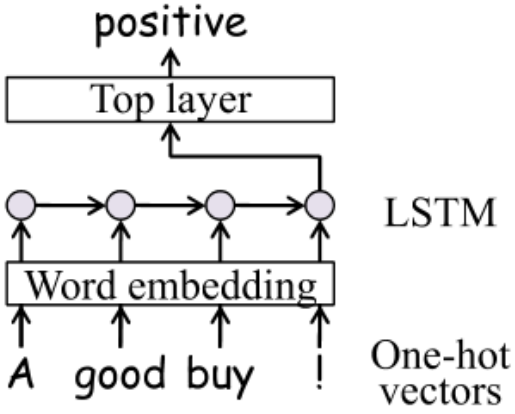

Рисунок 1.7 — Пример работы анализа тональности из работы [56].

Эти нелинейные функции называются гейтами. работа ячейки LSTM описана в уравнениях 1.16. Графическое представление этих уравнений можно увидеть на рис. 1.6а. Пример применения к задаче анализа тональности (классификации фраз по позитивной/негативной интонации, заложенной в них) можно увидеть на рис. 1.7.

$$
\begin{aligned}
\boldsymbol{c}'_t &= \tanh\left(W_{xc}\boldsymbol{x}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{c'}\right) && \textit{candidate cell state} \\
ß_t &= \sigma\left(W_{xi}\boldsymbol{x}_t + W_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) && \textit{input gate} \\
\boldsymbol{f}_t &= \sigma\left(W_{xf}\boldsymbol{x}_t + W_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) && \textit{forget gate} \\
ø_t &= \sigma\left(W_{xo}\boldsymbol{x}_t + W_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) && \textit{output gate} \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + ß_t \odot \boldsymbol{c}'_t, && \textit{cell state} \\
\boldsymbol{h}_t &= ø_t \odot \tanh(\boldsymbol{c}_t) && \textit{block output}
\end{aligned}
\tag{1.16}
$$

Ячейка LSTM имеет *входной гейт* (unput gate), *гейт забывания* (forget gate), и *выходной гейт* (output gate), в дополнение к собственно рекуррентной ячейке памяти и ее состоянию. Обозначая за $\boldsymbol{x}_t$ входной вектор в момент времени t; за $\boldsymbol{h}_t$ — вектор скрытого состояния в момент времени t; за $W_{x\cdot}$ (с различными вторыми нижними индексами) — матрица весов, применяемых к входному вектору; за $W_{h\cdot}$ — матрицы весов для рекуррентных взаимодействий, а за $\boldsymbol{b}$ — вектора смещения, получается следующее формальное определение: на шаге времени t, имея вход $\boldsymbol{x}_t$, предыдущее скрытое состояние $\boldsymbol{h}_{t-1}$, и состояние ячейки памяти $\boldsymbol{c}_{t-1}$, LSTM вычисляет $\boldsymbol{h}_t$.

Рекуррентная ячейка с гейтами

Рекуррентная ячейка с гейтами (Gated Recurrent Unit, GRU) — другая популярная архитектура, работающая в концепции запоминания состояния между шагами по времени. Она была представлена в работе Чо [51]. У GRU есть только два гейта — сброса и обновления:

- гейт сброса аналогичен гейту забывания в LSTM,
- гейт обновления можно рассматривать, как комбинацию входного и выходного гейтов LSTM.

Графическое представление GRU можно увидеть на рис. 1.6b. Более формально это описывается в уравнениях 1.17. Обозначая за $\boldsymbol{x}_t$ входной вектор в момент времени t; за $\boldsymbol{h}_t$ — вектор скрытого состояния в момент времени t; за $W_{x\cdot}$ (с различными вторыми нижними индексами) — матрица весов, применяемых к входному вектору; за $W_{h\cdot}$ — матрицы весов для рекуррентных взаимодействий, а за $\boldsymbol{b}$ — вектора смещения, получается следующее формальное определение: на шаге времени t, имея вход $\boldsymbol{x}_t$, предыдущее скрытое состояние $\boldsymbol{h}_{t-1}$, GRU вычисляет $\boldsymbol{h}_t$.

$$
\begin{aligned}
\boldsymbol{u}_t &= \sigma(W_{xu}\boldsymbol{x}_t + W_{hu}\boldsymbol{h}_{t-1} + \boldsymbol{b}_u),\\
\boldsymbol{r}_t &= \sigma(W_{xr}\boldsymbol{x}_t + W_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r),\\
\boldsymbol{h}'_t &= \tanh(W_{xh'}\boldsymbol{x}_t + W_{hh'}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})),\\
\boldsymbol{h}_t &= (1 - \boldsymbol{u}_t) \odot \boldsymbol{h}'_t + \boldsymbol{u}_t \odot \boldsymbol{h}_{t-1}.
\end{aligned}
\tag{1.17}
$$

Проведенные в последнее время сравнения, позволяют утверждать, что GPU обладают сходным с LSTM качеством в задачах моделирования последовательностей, как показано в работе Чуня [57], а учитывая, что GRU имеет меньше тренируемых параметров, эта архитектура становится все более и более популярной.

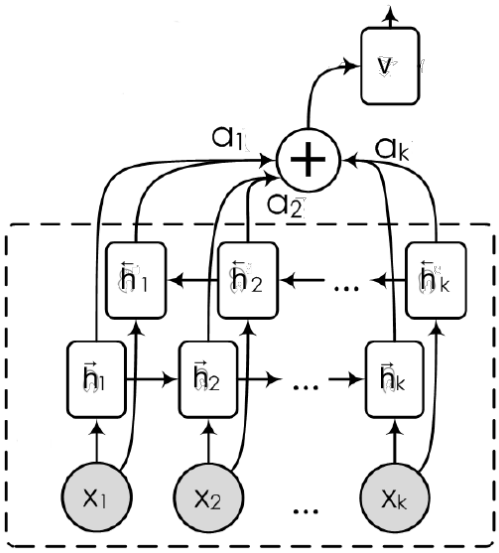

Рисунок 1.8 — Двунаправленная RNN с вниманием.

Механизм внимания

У рекуррентных нейронных сетей есть широко известный недостаток: они быстро забывают информацию с предыдущих шагов про времени, см. например работу Й. Бенжио [52]. Чтобы уменьшить этот негативный эффект был предложен механизм внимания, который сам по себе является некоторым типом памяти. Ставший уже классическим первый подход был предложен в работе Богданова [58]. Идея, лежащая в основе механизма внимания, может быть представлена, как выбор “наиболее интересной” части входной последовательности для генерации выхода на текущем шаге по времени. Модель мягкого выравнивания выдает веса α_{ti}, которые управляют тем, насколько каждое входное слово влияет на текущее выходное слово. Вес α показывает, должна ли сеть сосредоточиться на текущем слове прямо сейчас. υ — это текстовый вектор, который собирает в себе всю информацию из входных слов. Модель тренируется end-to-end: так как внимание является мягким (действительно-значной функцией), градиенты могут протекать через сеть. Мягкое внимание существенно улучшает перевод и классификацию для длинных предложений и на сегодняшний день является подходом по умолчанию. Более формально оно описано в уравнениях 1.18 и представлено на рис. 1.8.

$$\begin{aligned} \boldsymbol{\upsilon}_t &= \tanh\left(W_\omega \left[\overrightarrow{\boldsymbol{h}_t}, \overleftarrow{\boldsymbol{h}_t}\right] + \boldsymbol{b}_\omega\right) \\ \boldsymbol{\alpha}_t &= \frac{\exp\left(\boldsymbol{\upsilon}_t^T \boldsymbol{u}_i\right)}{\sum_{j=1}^{T} \exp\left(\boldsymbol{\upsilon}_j^T \boldsymbol{u}_i\right)} \\ \boldsymbol{\upsilon} &= \sum_{t=1}^{T} \boldsymbol{\alpha}_t \left[\overrightarrow{\boldsymbol{h}_t}, \overleftarrow{\boldsymbol{h}_t}\right] \end{aligned} \tag{1.18}$$

где W_ω и $\boldsymbol{b}_\omega$ параметры для линейроного преобразования скрытых представлений; $\boldsymbol{u}_i$ некоторый вектор, отличный от скрытых состояний, в случае LSTM это может быть вектор состояния ячейки памяти c_T в конце чтения последовательности. $\boldsymbol{u}_i$ может рассматриваться, как вектор контекста, в том смысле, что вектора, которые “ближе” (то есть имеют меньшее косинусное расстояние) к этому вектору должны иметь больший вес.

1.3.3 Сверточные нейронные сети

Другим распространенным подходом к построению нейронных сетей для обработки текстовых данных являются сверточные сети, пришедшие из задач компьютерного зрения. Сверточные нейронные сети были впервые упомянуты в работе Фукусимы [59] и основывались на идеях работы визуальной коры головного мозга.

Сверточная нейронная сеть обычно представляет собой чередование сверточных слоев (convolution layers), субдискретизирующих слоев (subsampling layers) и, часто, полносвязных слоев (fully-connected layer) на выходе. Все три вида слоев могут чередоваться в произвольном порядке, что подробно изложено в работе Чжана [36].

В сверточном слое нейроны, которые используют одни и те же веса, объединяются в карты признаков (feature maps), а каждый нейрон карты признаков связан с частью нейронов предыдущего слоя. При вычислении сети происходит свертка каждого нейрона с областью предыдущего слоя. Пример архитектуры сверточной нейронной сети изображен на рис. 1.9.

Рассмотрим более подробно каждый из элементов сверточной сети, которыми являются сверточные слои, слои субдискретизации и полносвязные слои.

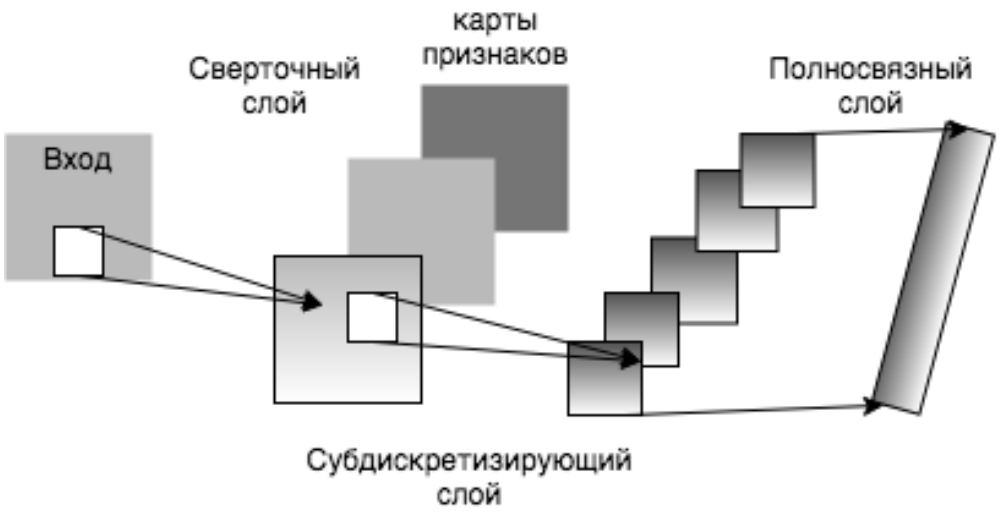

Рисунок 1.9 — Пример архитектуры сверточной сети

Полносвязные слои были рассмотрены в подразделе 1.3.1, другие два типа слоев будут рассмотрены ниже.

Сверточные слои

В сверточное слое нейрон не соединен со всеми нейронами предыдущего уровня, как в полносвязном слое, а только с некоторой областью предыдущего слоя, называемой “рецептивным полем”. Применяется матрица весов (ядро свертки) размера меньше, чем сам слой, ко всему обрабатываему слою. Окно ходит по большой матрице с некоторым-то шагом (stride). В целом свертка - это операция произведения Адамара двух матриц: части входной матрицы и ядра свертки. Для матрицы A свертку с ядром k можно произвести по следующей формуле:

$$b_{ij} = \sum \sum k \odot A_{[i-s_h, i+s_h]:[j-s_v:j+s_v]}, \tag{1.19}$$

где b_{ij} - это результат свертки.

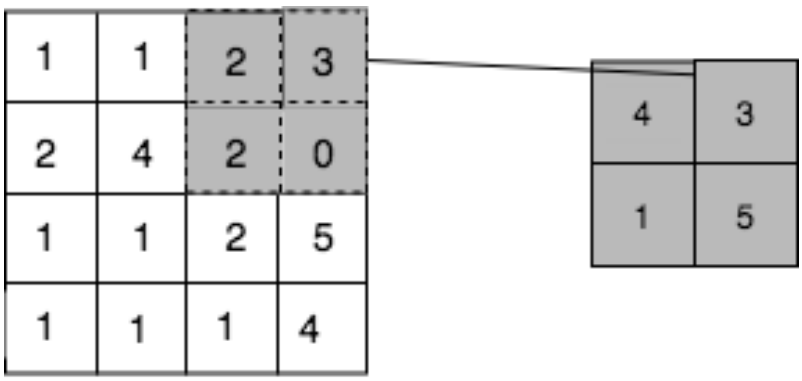

Рисунок 1.10 — Выбор наибольшего элемента с окном размера 2x2 и шагом 2

Субдискретизирующие слои

Слои данного типа служат для уменьшения размерности. Существует большое количество вариантов, как это сделать, но на практике самым распространенным является - выбор максимального элемента из блока/окна (max-pooling). Всю карту на выходе сверточного слоя рассматриваем в виде блоков, внутри которых и выбирается максимальный элемент. Пример использования субдискретизации с окном размера 2x2 с шагом 2 изображён на рис. 1.10.

Нелинейная функция ReLU

В сверточных слоях в качестве нелинейной функции часто используется так называемый Rectified Linear Unit (ReLU) - линейный фильтр, впервые представленный в работе Наира и Хинтона [47]. Математически он выражается так:

$$relu(x) = max(0,x), \tag{1.20}$$

где x - вход нелинейной функции. Важной особенностью это функции является то, что ее производная рассчитывается с помощью одной операции. Производная от функции $relu$ записывается следующим образом:

$$relu'(x) = \begin{cases} 1, & \text{если } x > 0. \\ 0, & \text{в обратном случае.} \end{cases} \tag{1.21}$$

1.4 Задача классификации текстов

Задача классификации текстов сводится к нахождению верной метки l для входного текста. Признаками текста являются слова $w_1, ..., w_n$. В зависимости от подхода, взаимное расположение слов может учитываться либо не учитываться. Таким образом, последовательности токенов ставится в соответствие метка. Классическим методом бинарной классификации текстов является TF-IDF представление входного текста, и использование полученного вектора в качестве вектора признаков для логистической регрессии.

Логистическая регрессия - метод бинарной классификации, основывающийся на предположении, что для входного объекта из позитивного класса вероятность позитивного класса должна быть равна 1, а из негативного класса соответственно 0.

$$P(y = 1|x) = \frac{1}{1 + e^{\theta^T x}}, \tag{1.22}$$

где y - метка класса, x - вектор признаков входного объекта, θ - набор обучаемых параметров логистической функции.

Более современные подходы используют векторное представление слов, описанное в 1.2, для того, чтобы получить больше информации из входного набора слов, т.к. векторные представления слов несут в себе закодированную семантику слов.

В работе Ю. Кима [60] векторное представление слов основано на векторном представлении символов. В работе показаны современные подходы к классификации, а также она обладает потенциалом по устойчивости в силу особенности описанной архитектуры, но этот аспект не был рассмотрен в рамках описываемой работы. В классической работе Винчьярелли из IDIAP [61] рассматриваются популярные на тот момент методы классификации текстов, например, SVM, но что более важно, в работе не рассматриваются методы *устойчивости* к шуму, а методы *коррекции* шума. То есть задача борьбы с шумом выносится на другой уровень рассмотрения. В работе Ц. Ниу и соавторов [7] исследуется нормализация медицинских концептов, которую можно рассматривать как классификацию на много классов. Авторы используют сверточные сети с вниманием на уровне символов, и их модель шума (точнее 4 вариации одной модели) близка по строению к той, которая используется в этой работе.

Многие работы по классификации текстов в наши дни опираются на векторные представления слов, так что полезно рассмотреть также этот аспект. Например, модель FastText была расширена механизмом классификации в работе Миколова и соавторов [25], эта модель классификации показывает хорошие результаты на чистых корпусах, но ее устойчивость к опечаткам ограничена. Эта устойчивость была исследована автором в работе [14].

В работе Ли [62] для анализа тональности была представлена модель, устойчивая к шуму, представленному выпадением части слов из входного предложения. В этой работе использовалась сверточная нейронная сеть, на которую накладывались различные регуляризаторы. Все неизвестные модели слова были представлены единственным токеном.

Существуют и ненейросетевые подходы к задаче классификации, например, Тутубалиной Е.В. [63], где авторы используют тематическое моделирование для улучшения классификации тональности.

1.5 Задача извлечения именованных сущностей

Задача извлечения именованных сущностей ставится следующим образом: на вход модели подается последовательность токенов $w_1, ..., w_n$, на выходе ожидается последовательность меток (тегов) $t_1, ..., t_n$. Самая распространенная схема меток (тегов) называется BIO-разметка. BIO-разметка состоит из трех типов меток: **B**egin - начальная метка сущности, **I**ntermediate - продолжающая метка сущности, и метка незначимого слова **O**ther. Пример:

> Vancouver/B-LOC is/O a/O coastal/O seaport/O city/O on/O the/O mainland/O of/O British/B-LOC Columbia/I-LOC. The/O city's/O mayor/O is/O Gregor/B-PER Robertson/I-PER.

В примере показаны два самых распространенных типа сущностей - географические наименования (LOC) и имена людей (PER). Существует множество разнообразных тегов, набор которых варьируется в зависимости от конкретной задачи. Например, корпус CAp'2017, рассмотренный в работе автора [12], содержит разметку по 13 различным сущностям, включающим в себя

по мимо упомянутых LOC и PER так разметку названий музыкальных групп и брендов.

Традиционно задача распознавания именованных сущностей решается с помощью *условных случайных полей* (Conditinal Random Fields, CRF), впервые описанных в работе Лафферти [64]. Условным случайным полем называют графовую модель, которая используется для представления совместных распределений набора нескольких случайных переменных. CRF состоит из следующих компонентов:

– неориентированный граф $G = (V, E)$, где каждая вершина $v \in V$ является случайной переменной и каждое ребро u,$v \in E$ представляет собой зависимость между случайными величинами u и v;
– набор потенциальных функций (potential function) φ_i.

Функция φ_i ставит каждому возможному состоянию набора элементов u_i в соответствие некоторое неотрицательное вещественнозначное число. На практике, как правило используются потенциальные функции для единичных элементов и их пар.

Вершины графа, не являющиеся смежными, соответствуют условно независимым случайным величинам.

Совместное распределение набора случайных величин $X = \{x_k\}$ в CRF вычисляется по формуле:

$$P(x) = \frac{1}{Z} \prod_k \varphi_k(x_k), \tag{1.23}$$

где $\varphi_k(x_k)$ – потенциальная функция, описывающая состояние случайных величин в k-ом наборе вершин; Z – коэффициент нормализации, вычисляемый по формуле:

$$Z = \sum_{x \in X} \prod_k \varphi_k(x_k) \tag{1.24}$$

Множество входных токенов $X = \{x_t\}$ и множество соответствующих им меток $Y = \{y_t\}$ в совокупности образуют множество случайных переменных $V = X \bigcup Y$. Потенциальные функции для задачи извлечения именованных сущностей задаются следующим образом:

$$\varphi_k = e^{\sum_k \lambda_k f_k(y_t, y_{t-1}, x_t)}, \tag{1.25}$$

где $\{\lambda_k\}$ - набор параметров модели, и $\{f_k(y_t, y_{t-1}, x_t)\}$ - набор признаковых функций. В настоящее время CRF как правило сочетаются с другими методами, прежде всего с рекуррентными нейронными сетями.

В работе Ниу [7] рассматривается задача нормализации медицинских концептов, которая может рассматриваться, как разновидность задачи извлечения именованных сущностей. В этой работе предложения модель, основанная на CNN. Важно отметить, что используемая модель шума аналогична, используемой в этой работе. Но в этой работе не была рассмотрена устойчивость к шуму, как отдельное свойство, а только продемонстировано само свойство для предложенной модели.

В работе Муна [65] рассматривается задача различения именованных сущностей с помощью дополнительных источников информации, таких, как изображения, которая является родственной к задаче извлечения именованных сущностей. В этой работе рассматривается естественный шум и способы борьбы с ним. В частности авторами была предложена модель *Deep Leveshtein*, близкая по архитектуре к модели, описанной в работе автора [11]. Предложенная модель демонстрирует устойчивость к шуму на предложенном датасете, но общая устойчивость к шуму также не была исследована в данной работе.

1.6 Задача извлечения аспектов

Задача извлечения аспектов ставится следующим образом: существует некоторый тип сущностей E, который полагается фиксированным. Этому типу сущностей приписывается некоторый набор аспектов A, который фиксирован относительно типа сущности. На вход модели подается текст, описывающий сущность типа E, на выходе модель должна выдать набор вещественнозначных чисел по количеству аспектов, интерпретируемых как вероятности или уверенности о наличии аспектов во входном тексте. Под аспектом как правило понимается некоторая существенная характеристика описываемого типа сущности. Пример:

У этого телефона громкий динамик.

В примере для типа сущностей “телефон” модель должна обнаружить наличие в это описании аспекта “динамик”.

Традиционно задача извлечения аспектов решается с помощью тематического моделирования. *Тематическое моделирование* - это задача построения тематической модели, то есть определения темы для каждого из документов коллекции. Пусть дана коллекция D и заранее определенный набор тем T, тогда задача тематического моделирования определить для каждого документа $d \in D$ набор тем $\{t\} \in T$, к которым относится этот документ. За время развития тематического моделирования было предложено несколько базовых подходов, таких как латентное семантическое индексирование, представленный в работе Пападимитриу [66], вероятностное латентное семантическое индексирование, представленный в работе Хофманна [67] и латентное размещение Дирихле, представленное в работе Бляя [68]. Эти модели являются графическими моделями, определившими основное направление развития этой области, однако в настоящее время начинают появляться работы, использующие нейронные сети для извлечения аспектов, например, работа Хе [69].

В работе Веселовской [70] авторы предлагают проверку орфографии в качестве шага предварительной обработки для задачи извлечения аспектов, которая основана на корпусной статистике. Ван и соавторы [71] также используют такой шаг предварительной обработки, но не указывают, какую систему они используют для исправления орфографии. Другой подход продемонстрирован в работе Паблос и др. [72] для майнинга мнений с извлечением аспектов - авторы прямо указали, что они игнорируют существующий шум в данных.

Все эти подходы имеют дело с шумом в данных основаны на предварительной обработке текста или в принципе игнорируют шум в данных, но сами описываемые модели не предназначены для борьбы с шумом.

1.7 Выводы к главе 1

Анализ предметной области показывает, что несмотря на существование большого количества работ, которые посвящены темам получения векторных представлений слов, классификации текстов и извлечению аспектов, очень мало из них затрагивают тему устойчивости к шуму, а полностью посвященные этой

проблеме работы отсутствуют. Что является упущением в силу наличия естественного шума в текстах, написанных пользователями, как показано в работе Кучержана [2], и неидеальности систем проверки орфографии, как показано в работе Сорокина [4]. Если при построении систем, решающих различные задачи естественного языка, заложить устойчивость к опечаткам, то можно превзойти по устойчивости существующие системы, опирающиеся на системы проверки орфографии, что было показано в работах автора [5; 12—14].

Глава 2. Устойчивые к шуму вектора слов

Данная глава посвящена сравнению программных систем векторных представлений слов, а также построению системы векторных представлений слов, которая устойчива к шуму. В отличие от систем описанных в разделе 1.2, программная система, которая будет рассматриваться в этой главе, не обучает векторные представления символов или функции для их объединения; ключевое отличие представляемой системы в том, что векторные представления слов не определены детерминистически. Векторные представления отдельных слов зависят от окружающих слов, следуя в этом смысле дистрибутивной гипотезе.

Целью данной главы является построение системы для получения векторных представлений устойчивых к различному виду шума - недостающие или избыточные буквы в словах и перестановки букв.

Для изучения общности предлагаемого подхода и сопоставления с существующими были выбраны языки из различных топологических групп, имеющих разную структуру аффиксов: английский как (почти) аналитический (грамматические отношения имеют тенденцию к передаче в основном через синтаксис, аффиксы слабо представлены в языке), русский как флективный (словоизменения происходят при помощи аффиксов/формантов, грамматические отношения передаются с помощью аффиксов/формантов, которые имеют тенденцию выражать сразу несколько грамматических значений) и турецкий - как агглютинативный (словоизменения за счет формантов, каждый из которых несет свое значение). Чтобы получить качественные результаты, тестирование проводилось для разных задач, таких как обнаружение парафраз, анализ тональности предложения (уровень тональности текста, показывающий эмоциональную окраску лексики) и распознавание логических связей между предложениями. Добавление шума к фразам из каждого набора данных позволяет нам сравнивать различные архитектуры на устойчивость к шуму. Для целостности результатов эксперименты проводились как для смеси шума, так и для каждого в отдельности.

Главным вкладом данной работы является сравнение существующих подходов для работы с зашумлёнными текстами и построение системы, позволяющей получать векторное представление для слов, устойчивое к описанным шумам.

Глава построена следующим образом: в подглаве 2.1 описывает постановка задачи построения устойчивых к шуму векторных представлений слов и метод построения устойчивых к шуму систем векторного представления слов. В подглаве 2.1 представлена постановка экспериментов и описание предлагаемого метода сравнения систем. В подглаве 2.7 – результаты экспериментов и их интерпретация.

2.1 Постановка задачи

В данной работе ставится цель сравнить существующие системы для получения векторных представлений слов и представить новую систему, демонстрирующую лучшее качество для слов содержащих опечатки. В рамках предлагаемой системы представления слов обучаются таким образом, чтобы они сохраняли семантический смысл исходных слов, чистых от шума. Способы измерения качества полученных векторов будут обсуждаться ниже.

Метод построения систем, устойчивых к шуму, в задаче векторного представления слов

Система векторного представления слов состоит из модуля побуквенного представления слов, модуля обработки контекста и процедуры обучения. Предлагаемый метод состоит в том, что:

- модифицируется модуль побуквенного представления слов таким образом, чтобы он был устойчив в опечаткам,
- модифицируется модуль обработки контекста для учета контекста слова,
- задается процедура обучения модуля контекстного представления на обучающей выборке.

Модуль обработки контекста обеспечивает сохранение семантических свойств при использовании модуля буквенного представления.

Разработанный метод использует два основных модуля: обработка структуры частей слова и контекстная зависимость слов получаемая посредством кодирования соседних слов. Гипотеза состоит в том, что данный метод позволит создавать инвариантные к шумам (типы шумов будут описаны в дальнейшем) векторные представления.

Для того, чтобы придать системе свойство устойчивости к шуму в рамках предлагаемого *метода создания программных систем* необходимо: а) выделить в системе компоненты буквенного представления слов и представления контекста, б) описать взаимодействие этих модулей и входных данных, в) описать процедуру обучения модуля контекстного представления на обучающей выборке. Модуль буквенного представления строится таким образом, чтобы быть мало- или нечувствительным к шуму разного рода. Модуль обработки контекста обеспечивает сохранение семантических свойств при использовании модуля буквенного представления.

Проверка предположения проводится посредством постановки экспериментов для различных языков и задач.

2.1.1 Описание задачи

Как упоминалось выше, одним из способов получения векторов слов является программная система Word2Vec [6]. Следующая задача стоит перед системами получения векторов слов. Пусть $\mathbf{S}$ – набор предложений. Каждое предложение $s \in \mathbf{S}$ состоит из слов $(w_1^s, w_2^s..w_{n_s}^d)$, где n_s - длина последовательности. Контекст слов $\mathbf{C}(w_i)$ - совокупность слов, отстоящих от данного не более, чем на k позиций, причем обычно k принимается от 2 до 12. $C(w_i) = \{w_{i-k}, w_{i-k+1}, ..., w_{i+k-1}, w_{i+k}\}$. Для построения векторных представлений слов обычно используется дистрибутивная гипотезой Харриса, приведенной в работе [73], согласно которой слова, встречающиеся в похожих контекстах, имеют семантически близкие значения. Задача построения векторных представлений заключается в сопоставлении каждому слову $w \in W$ вектора $v_w \in \mathbb{R}^n$, $m << |W|$. Основное требование к необходимому отображению — соответствие близким по смыслу словам близких по расстоянию векторов.

Простейший вариант отображения слов в вектора называется one-hot кодирование. Длина вектора в таком случае будет равна размеру словаря, а сама информация в векторе не сможет отобразить ни связи между контекстами, ни связи самих слов. В большинстве систем, для каждого слова строится два векторных представления. Пусть $\mathbf{V}$, $\mathbf{U} \in \mathbb{R}^{m \times |W|}$ матрицы, столбцы которых соответствуют векторам слов и слов связанных с контекстом соответственно.

2.2 Векторные представления слов

Одна из первых идей получения векторов для слов, учитывая контекстную зависимость, была предложена в работе Бенжио [74]. Авторы предлагают рассмотреть задачу моделирования языка по контексту слева.

$$p(w_1, ..w_n) = \prod_{i=1}^{N} p(w_n|w_{n-1}, ...,w_1) \approx \prod_{i=1}^{N} p(w_n|w_{n-1}, ...,w_{n-k}) \tag{2.1}$$

где N - длина текста. Предложенная нейронная сеть принимает на вход k слов $w_1..w_k$ в формате one-hot кодирования, ставит им в соответствие их векторные представления и передает на вход следующему слою. Выходом такой сети является дискретное распределение над множеством всех слов словаря. В следующей главе будет подробнее описаны нейронные сети.

2.2.1 Векторные представления слов на уровне символов

В главе 1.2 уже был описан основной подход для работы с векторами слов. Основная проблема систем, основанных на работах Т. Миколова [25] и [6] - это слова, которых нет в словаре. Кроме того такие представления не дают возможности работать с зашумлёнными текстами, так как слова с опечатками являются OOV.

Все эти проблемы приводят к идее посимвольных представлений (character-level representations). Первые попытки учесть структуру слова состояли в том, чтобы попытаться разложить слово на морфемы, наименьшие

смысловые части письменного языка, как в работе Бота [75]. Если бы морфемы были доступны напрямую, они действительно стали бы подходящим вариантом для низкоуровневых представлений слов. Однако на самом деле получение морфем - это отдельная задача, которая также решается не со 100% качеством, как это показано в работе Выломовой [35].

Поэтому сейчас основные посимвольные системы получения представлений для слов работают на уровне букв. В работе Вана [76] представлена базовая система C2W (character to word), которая может получать представления слов из составляющих их букв с помощью двунаправленных LSTM, которые были описаны в разделе 1.3.2. В этой системе поданные на вход символы сначала преобразуются в соответствующие им векторы, а затем векторы подаются в двунаправленный LSTM, который уже выдает представление слова.

Еще одним интересным подходом к получению векторных представления является использование одномерных сверточных сетей над one-hot представлениями. Этот подход отлично зарекомендовал себя как на уровне символов, так и на уровне букв. Данные системы активно используются в задачах обработки естественного языка.

2.2.2 Описание используемых подходов к обработке текстов

В данном разделе будут описаны основные подходы из нейронных сетей, которые будут использованы в дальнейшем в этой главе и которые не были описаны в главе 1.

2.2.3 Ячейка нейронной сети Simple Recurrent Unit

Основным недостатком описанного в 1.3.2 подхода является его скорость работы. Рекуррентные нейронные сети являются немасштабируемыми для матричных вычислений методами, так как содержат связи между моментами времени, то есть должны вычислять последовательно. На данный момент существует множество работ посвященных ускорению RNN, например, работы

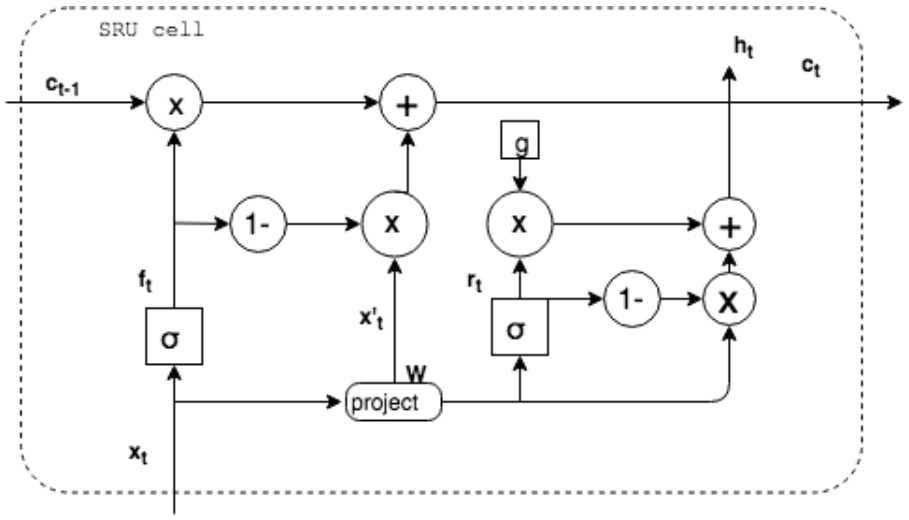

Рисунок 2.1 — Граф вычисления для ячейки SRU.

Миколова [77] и Кутника [78], в данной работе будет использоваться одно из таких улучшений под названием простой рекуррентный элемент (SRU, Simple Recurrent Unit), описанное в работе [79]. Диаграмма одной ячейки SRU представлена на рис. 2.1. Основной вклад, добавленный SRU - это избавление от рекуррентной зависимости при вычислении h_t. Система, описывающая одну итерацию, приведена ниже:

$$
\begin{aligned}
\mathbf{x}'_t &= \mathbf{W} x_t \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f * x_t + b^f) \\
\mathbf{r}_t &= \sigma(\mathbf{W}^r * x_t + b^r) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - f_t) \odot \mathbf{x}'_t \\
\mathbf{h}_t &= \mathbf{r}_t \odot \mathbf{g}(c_t) + (1 - r_t) \odot \mathbf{x}_t
\end{aligned}
\tag{2.2}
$$

здесь σ – сигмоидальная функция, $\mathbf{x_t}$ – входной вектор, $\mathbf{x'_t}$ – линейно-преобразованный входной вектор, $\mathbf{W^f}, \mathbf{P^r}$ матрицы рекуррентных весов, $\mathbf{r}$ и $\mathbf{f}$ обозначают гейты сброса и памяти соответственно и $\mathbf{c}$ обозначает ячейку памяти.

Данный подход позволил ускорить обучение RNN в десятки раз, дав возможность использовать механизмы для распараллеливания вычислений. На данный момент это одна из самых быстрых модификации рекуррентных нейронных сетей. В данном случае видно, что в системе 2.2 можно параллельно вычислять первые три уравнения, в то время как в ячейке LSTM рекуррентная зависимость проявляется в каждом из уравнений 1.16. Перенос рекуррентной зависимости в последние уравнения позволил не только увеличить скорость обучения, но и поднять качество на некоторых задачах.

2.3 Разработанная система: RoVe

В данном разделе описывается предложенная система для построения векторных представлений. Система была предложена в работе [11], названа она так по первых слогам словосочетания “robust vectors”, то есть устойчивые вектора [слов]. Система RoVe сочетает в себе контекстную зависимость и открытую лексику, что позволяет создавать осмысленные векторные представления для слов OOV. Эти две функции поддерживаются двумя частями системы.

2.3.1 Представление слов BME

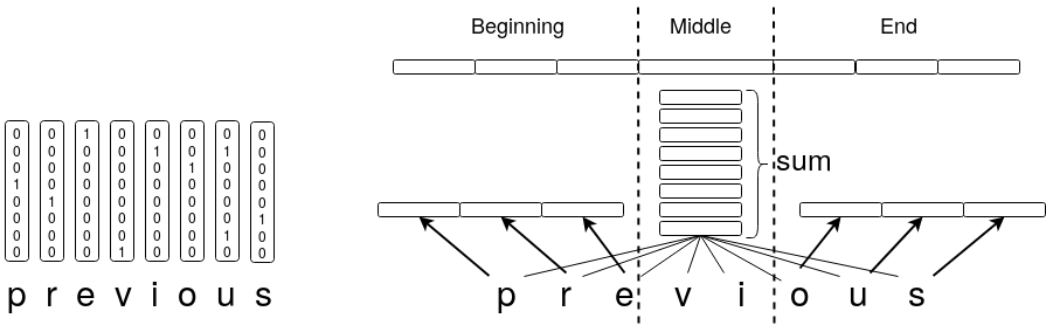

Рисунок 2.2 — Генерация вектора для слова *previous*. Слева: генерация векторов one-hot кодирования букв, справа: генерация представления BME.

Для достижения устойчивости системы используется представление BME для слов, предложенное в работе [11]. Ниже идет подробное описание, работы этого представления, обобщенное для случая произвольных параметров.

Во-первых, каждый символ слова представляется, как one-hot вектор (вектор нулей размером в длину алфавита с одной 1 в позиции i, где i - индекс символа). Тогда имеются три вектора: начальный ($\mathbf{B}$), средний ($\mathbf{M}$), и конечный ($\mathbf{E}$)[1] вектор, которые конкатенируются вместе. Вектор $\mathbf{M}$ - это сумма one-hot векторов всех символов слова. $\mathbf{B}$ - представляет собой объединение в one-hot вектора n_b первых букв слова. Аналогично, $\mathbf{E}$ компонент представляет собой объединение в один one-hot вектор n_e последних символов слова. n_b и n_e являются гиперпараметрами, которые могут различаться для разных наборов данных.

[1]Представление названо по этим векторам.

Входное представление слова формируется путем объединения **B**, **M** И **E** векторов. Поэтому его длина равна $(n_b + n_e + 1) \times A$, где A - алфавит языка. Этот входное представление далее обрабатывается нейронной сетью, описанной ниже. Формирование входного вектора показана на рис. 2.2.

Пусть слово w состоит из набора букв. Обозначим за $c_1..c_k$ one-hot представления этих букв. Тогда

$$B(w) = c_1\|..\|c_{n_b} \tag{2.3}$$

$$E(w) = c_{k-n_e}\|..\|c_k \tag{2.4}$$

$$M(w) = \sum_{1}^{k} c_i \tag{2.5}$$

$$BME(w) = B(w)\|M(w)\|E(w) \tag{2.6}$$

где $\|$ обозначает операцию конкатенации. Представление BME было вдохновлено работой Сакагучи и соавторов [80], где первый и последний символы слова кодируются отдельно, поскольку они несут больше смысла, чем другие символы. Стоит упомянуть также представление `meanChar` из работы Белинкова и Биска [8]. Это представление формируется, как усреднение векторных представлений символов, выученных в процессе обучения системы машинного перевода.

Однако мотивация для представления BME проистекает из разделения слов на морфемы. Кодирование n_b первых символы и n_e последних символов слова в фиксированном порядке (в отличие от остального слова, которое сохраняется как мешок букв) основывается на предположении, что это может быть аффикс, который несет в себе определенное значение (например английский префикс *'un'* со значением обратного действия или отсутствия) или грамматической информации (например, суффикс *'ed'*, который указывает на глагол прошедшего времени). Таким образом, сохранение его может сделать полученное векторное представление более информативным.

Часть представления, обозначаемая M, отбрасывает порядок букв в слове. Это гарантирует устойчивость векторных представлений против перемены мест соседних букв в слове, что является одной из наиболее распространенных опечаток. Сравнивая *information* и *infromation*, можно отметить, что эти слова будут иметь одинаковые векторные представления в предлагаемой системе, в то время как Word2Vec и многие другие системы не смогут обеспечить какое-либо представление для последнего слова.

В дополнение к этому, представление BME не ограничено никаким словарем и может обеспечить векторное представление для любого слова, включая слова с опечатками. Кроме того, если слово с ошибками достаточно близко к его первоначальной версии, его представление также будет близко к исходному слову. Поэтому даже неправильно написанное слово, скорее всего, будет истолковано правильно.

Использование предлагаемой системы снимает необходимость в орфографической коррекции, потому что слово не должно быть написано правильно, чтобы быть успешно интерпретированным. Как показано в разделе 2.7, орфографическая коррекция не обеспечивает должного уровня устойчивости к шуму. В отличие от других систем, которые не справляются с опечатками в словах, RoVe может обрабатывать шум, как в процессе обучения, так и в процессе тестирования.

2.3.2 Архитектура системы

Как уже было сказано выше, одним из основных модулей системы RoVe является BME представление. Выход из модуля побуквенного представления BME подается на модуль обработки контекста (см. рис. 2.3), который кодирует контекст в векторное представление с помощью нейронной сети.

Система RoVe создает контекстно-зависимые представления слов. Это означает, что она не генерирует фиксированный вектор для слова и должна производить его с нуля для каждого слова. Эта структура незначительно увеличивает время обработки текста, но дает более точные представления с учетом контекста. Система концептуально аналогична кодировщику, используемому для создания представления предложения путем чтения всех его слов. Такие кодировщики используются в машинном переводе [40], вопросно-ответных системах [81] и многих других задачах.

Для того, чтобы создать представление слова, необходимо кодировать его вместе с его контекстом. Для каждого слова контекста сначала создается его входное векторное представление (описано в разделе 2.3.1). Это представление затем передается кодировщику (верхняя часть рис. 2.3), который обрабатывает все слова контекста.

Кодировщик должен быть нейронной сетью, которая может обрабатывать последовательность слов и хранить информацию об их контекстах. Наиболее очевидным выбором является RNN или CNN, описанные в главе 1. Однако можно использовать и другие типы нейронных сетей. После обработки всего контекста получается векторное представление для целевого слова, передавая скрытое состояние, соответствующее нужному слову, через полностью связанный слой. Поэтому можно генерировать векторные представления для всех слов в контексте одновременно.

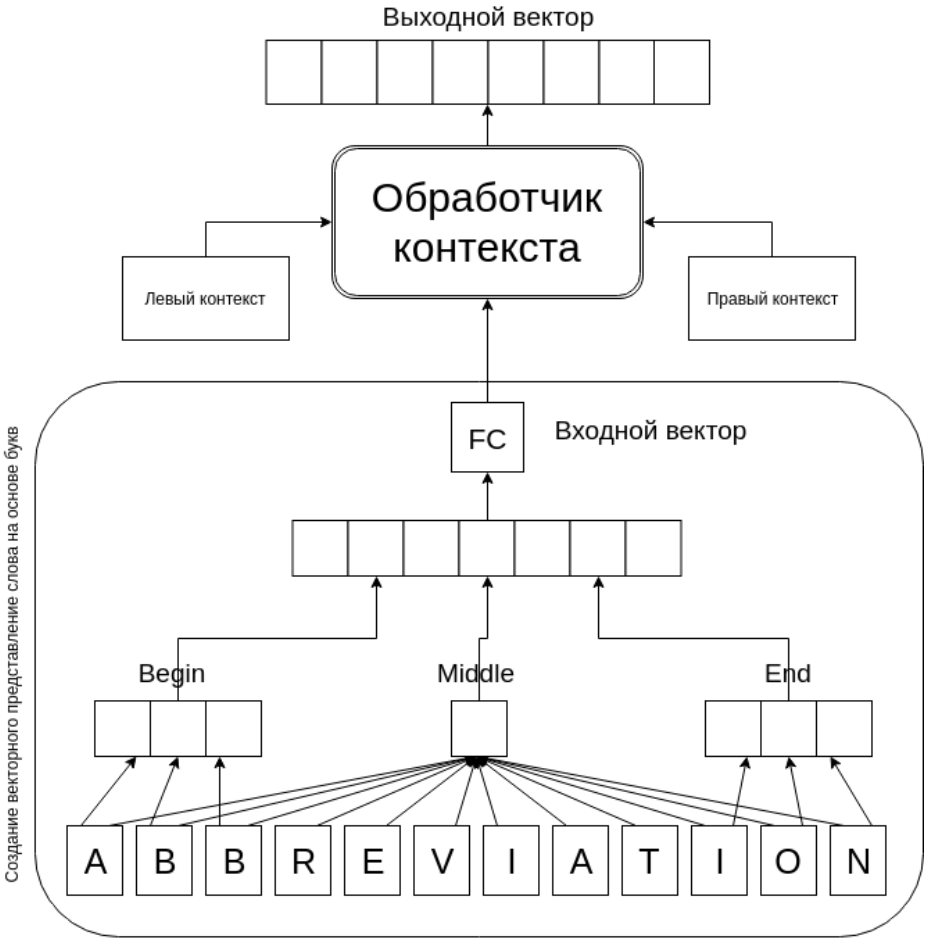

Рисунок 2.3 — Система RoVe: генерация векторного представления слова *abbreviation*.

$$RoVe(w) = enc(BME(w); C_{left}, C_{right})$$

где C_{left} и C_{right} обозначают левый и правый контексты для слова w соответственно.

Таким образом, вектора, генерируемые системой, всегда зависят от контекста. Контекст может быть полным предложением или окном вокруг слова. Эта система не может порождать вектора для изолированных слов (с формальной точки зрения изолированное слово является словом с контекстом нулевой длины и в данной интерпретации предлагаемая система способна породить векторное представление слова). Однако конкретно такая задача сравнения изолированных слов возникает не часто, так как на практике типично приходится обрабатывать слова, имеющие определенный контекст.

Для обучения данной системы использован подход сэмплирования негативных примеров и функционал 1.5. Причем можно заметить, что скалярные произведения представлений слов, которые совпадают с косинусной мерой близости 1.6, в случае если вектора нормированы. Поскольку косинусная мера близости и скалярное произведение монотонно связаны между собой, то между оптимизацией таких функций есть связь.

Данный факт обосновывает возможность применения косинусной меры в качестве меры близости векторных представлений.

Обучается такая система с помощью алгоритма стохастической оптимизации Adam, описанного в статье Кингма и Ба [82]. Данный метод позволяет сходиться к минимуму быстрее, чем упомянутый в разделе 2.4 стохастический градиентный спуск. Размер пространства оптимизируемых параметров напрямую зависит от типа кодировщика, различные варианты которого обсуждаются ниже.

2.4 Исследуемые варианты системы RoVe

Так как выше описана довольно обобщенно система предложенная для решения поставленной задачи, в этом разделе будут рассматриваться различные кодировщики, сделанные на основе элементов описанных в разделе 1. В дальнейшем именно эти системы используются для проведения экспериментов. В случае несовпадения с таблицами результатов названия исследуемых систем указаны в скобках.

Стоит лишний раз подчеркнуть, что все представленные системы (кроме, разумеется, базовых) отличаются именно кодировщиком.

2.4.1 Системы с рекуррентными кодировщиками

– SRU,
– двунаправленная SRU (biSRU),
– LSTM,
– двунаправленная LSTM (biLSTM),
– stackedLSTM.

Первые четыре системы, основаны на подходах, подробно описанных в 1, и используют одноименный с подходом кодировщик.

Необходимо более подробно описать последнюю системы, которая не была описана в главе 1.

Основным ограничением двунаправленной архитектуры является то, что она не позволяет обмениваться прямой и обратной информацией между слоями. Архитектура составленных друг с другом LSTM (stackedLSTM), устраняет эту проблему. stackedLSTM позволяет комбинировать выходные данные прямого и обратного прохода по данным и использовать их в качестве входных данных для следующего слоя. Модуль обработки контекста в этой системе представлен двумя двунаправленными слоями с ячейками LSTM. Данная система обучается дольше, чем biLSTM, но чаще показывает хорошие результаты в различных задачах обработки естественного языка.

2.4.2 Системы, основанные на сверточном кодировщике

Существует несколько вариантов архитектуры сверточных сетей, которые эффективны при моделировании последовательностей слов. Основная идея заключается в использовании сверточных слоев вместо рекуррентных и передаче смысла из контекста посредством таких преобразований. Данные системы часто применяются в задачах по обработке естественного языка, что заставляет посмотреть в их сторону, при выборе наилучшего кодировщика.

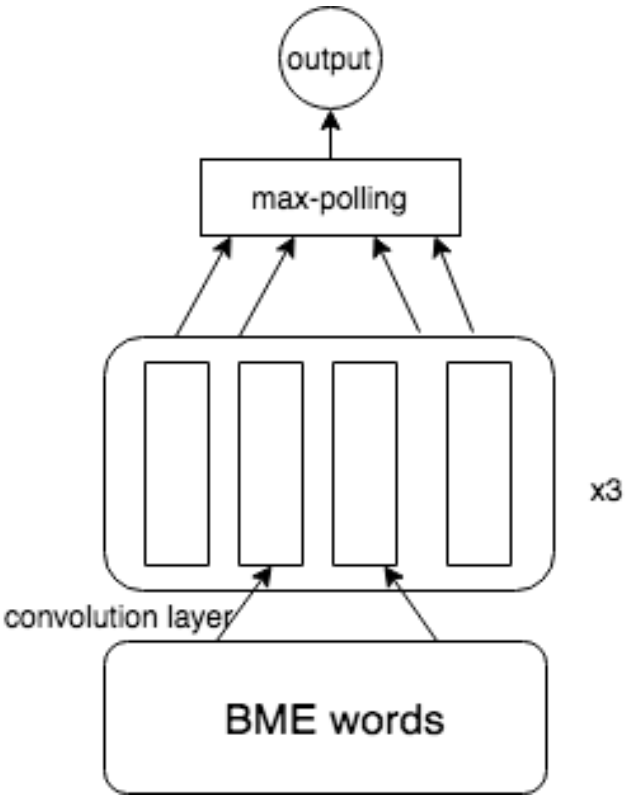

Рисунок 2.4 — Система на основе CNN.

Система CNN-1d

CNN-1d - сверточная нейронная сеть, которая применяется к матрице BME представлений по одному измерению, откуда и происходит название.

Эта система использует два полносвязных слоя в качестве отображения и 3 сверточных слоев с размерами ядра 3, 5 и 3 соответственно в качестве модуля обработки контекста.

Векторное представление Y может быть определено, как: $y_i = relu(w \times x_{i:i+n-1} + b)$, где w и b параметры и $x_i i : i + n - 1$ - это столбец матрицы, которая содержит вектора из x_i и $x_i i + n - 1$, где $x_i \in \mathbb{R}^D$ - это вектор-столбец.

В качестве нелинейной функции в сверточном слое используется ReLU. После последнего сверточного слоя следует субдискретизирующий слой max-pooling. Графическое представление можно найти на рис. 2.4.

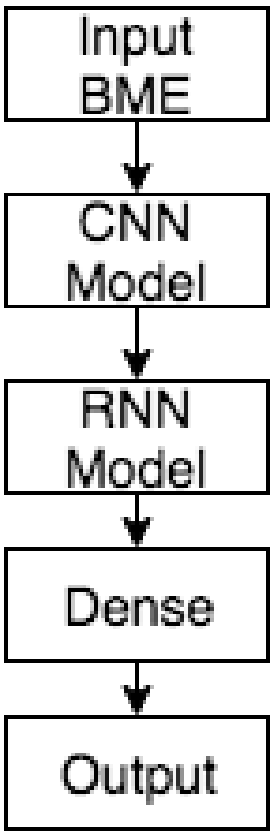

Рисунок 2.5 — Система ConvLSTM

Система ConvLSTM

Сверточные слои и рекуррентные слови могут быть объединены в рамках одной системы, чтобы получить более точные результаты в описываемой задаче. В начале применяется свертоный слой для входных векторов в представлении BME, примером такой системы может служить система **cnn-1d**.

Далее полученные вектора обрабатываются как в системе **biSRU**. В этом подходе ставится цель получить систему, которая строит языковую модель. На рис. 2.5 имеется иллюстрация архитектуры системы.

2.5 Наборы данных

В данном случае удобно разделить датасеты по языкам, соответственно присутствуют три группы.

Ниже приведен полный список используемых данных с соответствующей ему задаче.

– Английский: определение перефразировок: Microsoft Research Paraphrase Corpus, описанный в работе [83], определение логической связи: Stanford Natural Language Inference Corpus, описанный в работе [84], анализ тональности: Stanford Sentiment Treebank, описанный в работе [85];
– Русский: определение перефразировок: Russian Paraphrase Corpus, описанный в работе [86], анализ тональности: Russian Twitter Sentiment Corpus, описанный в работе [87];
– Турецкий: определение перефразировок: Turkish Paraphrase Corpus (TPC), описан в работе [88];

Microsoft Research Paraphrase Corpus [2] состоит из пар предложений, которые были извлечены из новостных источников в сети и проаннотированы специалистами. Аннотация была произведена двумя аннотаторами и дополнительно в случаях, когда оценки расходились, решение принимал третий специалист. Разметка подразумевает, что каждой паре сопоставляется число 0 или 1, которое обозначает является ли данная пара парафразом (т.е. эквивалентны ли оба предложения по смыслу). Корпус Microsoft Research Paraphrase Corpus состоит из 5.801 пары предложений, разбитых на обучающую (4.076 пар предложений, 2.753 пар в позитивном классе, т.е. точных парафраз) и тестовую выборки (1.725 пар предложений, 1.147 в позитивном классе).

Stanford Sentiment Treebank - набор данных, представленный группой Стенфордского университета, содержит в себе 11.855 предложений, каждому из которых сопоставлен уровень тональности: позитивный, нейтральный или негативный (в рамках данного исследования нейтральный класс был убран из данных, поскольку не несет полезной информации для задачи, как показано в работе Лукашевич и соавторов [89]). Всего позитивно и негативно окрашенных предложений в корпусе 10.662, поровну разделенных между этими классами. Каждое предложение было размечено тремя аннотаторами. Для этого набора данных опубликовано разделение на обучающую, валидационную и тестовую выборки.

[2] Доступен для скачивания по ссылке: https://www.microsoft.com/en-us/download/details.aspx?id=52398

Таблица 1 — Распределение оценок в датасете SNLI

класс / выборка	**логическое следование**	**нейтральный**	**противоречие**	**-**
тренировочная	183.416	182.764	183.187	785
валидационная	3.329	3.235	3.278	158
тестовая	3.368	3.219	3.237	176

Stanford Natural Language Inference - набор данных, состоящий из 570.152 пар предложений, извлеченных из подписей к фотографиям на сервисе Flickr, размеченных 5 людьми каждая. Этот корпус также состоит из трех классов, один из которых был удален из данных, для упрощения методов сравнения. Оставшиеся классы: противоречие (когда предложение 2 противоречит по смыслу предложению 1) и логическое следование. В этом корпусе все классы примерно равны по мощности. Более подробная статистика по набору данных представлена в таблице 1.

Корпус Russian Twitter Sentiment был собран автоматически из русского сегмента сети Twitter. Отбор кандидатов производился по наличию в них эмоджи, которые полагались признаком наличия в них тональности. Для включения в корпус были отобраны только те сообщения, которые не содержали противоречивых эмоджи, то есть эмоджи, указывающих на разную тональность. Корпус состоит из 114.911 позитивных и 111.923 негативных записей. Записями в данном случае являются отдельные твиты. Записи из сети Твиттер характеризуются наличием в них естественного шума.

Корпус Paraphraser содержит новостные заголовки на русском языке от различных информационных агентств, которые предполагаются (с помощью автоматической системы оценивания) близкими по смысловому значению. Кроме того, все они протестированы, чтобы быть близкими во время создания. Корпус содержит около 6.000 пар фраз, которые обозначены как -1 - не парафраза, 0-слабая парафраза и 1 - сильная парафраза. Для оценки были взяты только классы -1 и 1, т. е. не парафраз и сильный парафраз. Таких пар в корпусе 4.470.

Для турецкого языка был использован лишь один корпус из-за отсутствия других публично доступных данных по этому языку. **Turkish Paraphrase Corpus** (TPC)., точнее его публично доступная часть, также составлена из заголовков документов из новостного домена[3]. Эта часть содержит 846 пар

[3]TPC доступен для скачивания по ссылке: https://osf.io/wp83a/

предложений из новостных источников в интернете и снабжена аннотациями, указывающими, фиксирует ли каждая пара отношение парафразы/семантической эквивалентности.

К сожалению, другие рассматриваемые в этой главе задачи, а именно, определение логической связи и анализ тональности, не имеют общедоступных наборов данных на этом языке.

2.6 Эксперименты на прикладных задачах обработки текстов

В данном разделе подробно описана постановка эксперимента, а именно задачи, данные и системы. Дается разъяснение по используемым в качестве базовых системам, с которыми происходит сравнение предлагаемого подхода. Кроме того приводятся подробные результаты и их анализ. Дополнительно проводится исследование моделируемого шума.

2.6.1 Метод сравнения систем векторных представлений слов

Метод сравнения программных систем на предмет устойчивости к шуму состоит из следующих этапов: а) этапа контролируемого внесения шума во входные данные, б) этапа проверки качества тестируемой системы. Для того, чтобы проверить качество некоторой системы при заданном уровне шума необходимо внести во входные данные для тестирования системы шум заданного уровня и проверить качество вывода системы относительно известной разметки для тестовых данных. Данную процедуру необходимо провести для каждой системы и каждого уровня шума. Каждый такого рода эксперимент необходимо повторить некоторое количество раз для статистической достоверности получаемых результатов. В данной главе каждый эксперимент был повторен 10 раз, полученные результаты имеют достоверность $p < 0.05$.

В качестве метода сравнения систем предлагается метод внесения шума в тестовые данные. Уровень шума варьируется от 0% до 30%. Для сравнения си-

стемам, обученным на чистых, незашумленных данных подается на вход текст с внесенным шумом и замеряется качество на некоторой задаче.

Системы тестируются с различными уровнями шума от 0% (без шума) до 30%. По данным из работы Кучержана и Брилла [2], реальный уровень шума в пользовательских текстах составляет 10-15%.

В работе проведено моделирование трех различных типов шума:

1. добавление символа из словаря,
2. удаление символа из слова,
3. перестановка соседних символов.

Каждый из типов шумов накладывался независимо на буквы в слове с заданной вероятностью p - общий уровень зашумленности текстовых данных.

Для каждого вводимого слова случайным образом вставляется или удаляется буква с заданной вероятностью. Три типа шума вводятся одновременно для экспериментов, так как реальные ошибки встречаются сразу все вместе. Шум добавляется только к данным для специфичных задач, таких как классификация текстов или обнаружение парафраз, системы RoVe, Word2Vec и fastText обучаются на чистых данных.

2.6.2 Используемые для сравнения задачи

Качество векторов сгенерированных RoVe проверяется на различных задачах:

1. определение парафраз,
2. анализ тональности текста,
3. определение логической связи.

Для всех задач обучаются базовые системы, основанные на популярных вариантах архитектур векторных представлений слов. Для задачи анализа тональности обучается наивный байесовский классификатор, где в качестве признаков используются вектора слов. Перефразирование и логическое связывание выявляются путем вычисления косинусной меры схожести между векторами соответствующих предложений. Это сделано намеренно, чтобы убедиться, что качество в значительной степени определяется качеством векторов, которые используются, а не сложностью выучиваемых системами зависимостей. Для всех

задач сравниваются вектора слов, порожденные различными модификациями RoVe, с векторами, порожденными системами Word2Vec и fastText, так как в главе 1.2 обсуждалось, что это наиболее широко используемые подходы для получения векторных представлений для слов.

Дополнительно, чтобы показать, что предлагаемая система RoVe способна заменить в реальном использовании векторные представления fastText, которые в данный момент используются шире, Word2Vec, т.к. показывают более высокое качество, что также можно пронаблюдать по результатам проведенных экспериментов, были проведены эксперименты, в которых при проверке данные подавались на вход этих cbcntv после проверки орфографии.

2.6.3 Постановка экспериментов

Эксперименты проводятся с наборами данных для трех языков: английского (аналитический изолирующий язык), русского (синтетический флективный) и турецкого (синтетический агглютинативный). Аффиксы имеют различные структуры и цели в этих типах языков, и в экспериментах показывается, что представление BME эффективно для всех из них. Параметры BME не настраивались (n_b и n_e, длины B и E сегментов BME). Во всех экспериментах они выбираются равными 3, следуя тому факту, что средняя длина аффиксов на русском языке равна 2.54 [90]. Однако, вероятно, они не являются оптимальными для английского и турецкого языков.

2.6.4 Метрики оценки качества для тестируемых задач

В данной работе как было сказано выше качество обученных векторных представлений оценивалось на трех задачах:

1. определение парафраз,
2. анализ тональности текста,
3. определение логической связи между предложениями.

Для каждой из них решалась задача бинарная классификация: требуется отнести объект из выборки к одному из двух классов. Такую задачу в случае несложной структуры данных можно хорошо решать с помощью описанного ниже в этой главе метода: наивного Байесовского классификатора. Так было сделано в задаче анализа тональности. Для задач определения парафраз и логической связи использовался еще более простой метод сравнения.

Имея два векторных представления: x_1 и x_2 соответствующие предложению 1 и его парафразу. Тогда вероятность того, что x_1 совпадает с x_2 равна косинусной мере близости между векторами $cos(x_1,x_2)$, определенной уравнением 1.6.

Для оценки качества такой классификации удобно использовать метрику под названием ROC-AUC (ROC - receiver operating characteristic, “кривая ошибок”, AUC - area under curve, площадь под “кривой ошибок”). ROC-AUC – площадь под ROC-кривой – часто используют для оценивания качества упорядочивания алгоритмом объектов двух классов, значение лежит на отрезке [0, 1]. В случае идеального классификатора результат ROC-AUC будет равен 1. При случайном предсказании алгоритма классификации ROC-AUC = 0.5.

2.6.5 Наивный Байесовский классификатор

В задаче классификации необходимо определить класс по известному вектору признаков. В описываемом случае таким вектором является векторное представление слова. Байесовский классификатор выбирает класс, для которого максимальна вероятность при условии вектора признаков.

$$a(x) = argmax_{class} p(class|x) \tag{2.7}$$

2.6.6 Обучение систем

Системы RoVe обучались на следующих данных:

– Английский — корпус Reuters [91],

– Русский — Национальный корпус русского языка [92],
– Турецкий — корпус “42 bin haber”.

Все системы RoVe обучаются на исходных, неизмененных корпусах без добавления шума или какой-либо другой предварительной обработки.

Системы Word2Vec и fastText обучались на тех же данных, что и системы RoVe. Так как качество этих систем в экспериментах было низким, дополнительно были использованы общедоступные системы Word2Vec для русского и английского языков.

Помимо этого для системы fastText была проведена серия экспериментов, где на вход этой системе подавались данные после проверки системой проверки орфографии. Для проверки орфографии была использована система Yandex.Speller, которая предоставляет общедоступное API для проверки орфографии для русского и английского языков. Для турецкого языка проверка орфографии не проводилась.

Список базовых систем, которые используются в экспериментах:

– **Word2Vec** - общедоступная система векторных представлений Word2Vec;
– **fasttext** - общедоступная система векторных представлений fasttext;
– **fasttext+speller** - общедоступная система векторных представлений fastText, данные для которой обрабатываются системой проверки орфографии перед подачей на вход системе векторного представления слов;
– **C-Word2Vec** - система векторных представлений Word2Vec, обученная на том же корпусе, что и система RoVe;
– **C-fasttext** - система векторных представлений fasttext, обученная на том же корпусе, что и система RoVe;
– **C-fasttext+speller** - система векторных представлений fasttext, обученная на том же корпусе, что и система RoVe, и данные для которой обрабатываются системой проверки орфографии перед подачей на вход системе векторного представления слов.

2.7 Результаты экспериментов для векторных представлений слов

Ниже приведены результаты основных экспериментов.

В качестве системы fastText используются официальные предварительно обученные системы fastText[4]. Для корректного сравнения в условиях приближенных к реальному использованию используется расширенная версия системы fastText для русского и английского – fastText+speller. Для всех рассматриваемых задач тексты предобрабатываются общедоступной системой проверки орфографии Yandex.Speller перед получением векторов слов. Поскольку проверка орфографии является одним из распространенных способов уменьшения эффекта опечаток, это позволяет провести наиболее честное сравнение качества базовых системой с предлагаемой системой RoVe.

Чтобы не загромождать нижеследующие разделы таблицами все результаты приведены для смешанного шума.

2.7.1 Эксперименты для английского языка

Для английского языка будут представлены результаты экспериментов для всех типов задач, перечисленных выше в разделе 3.

В качестве базового подхода используется система Word2Vec, обученная на корпусе Google News и доступную без ограничений в интернете[5]. Это одна из крупнейших доступных систем для английского языка. Она была обучена на корпусе из 3 миллиардов слов, имееет 3 миллиона токенов и содержит вектора с размерностью 300, как в предлагаемой системе. В отличие от предлагаемого подхода, для этой системы необходимо использовать лемматизацию.

Эксперименты на данных MRPC

Результаты представлены в таблице 1. По этой таблице можно сделать следующие выводы. Самые важные результаты приведены в первых четырех колонках, так как они отражает уровень шума в тексте до 15% включительно, то

[4]для английского языка: https://fastText.cc/docs/en/english-vectors.html,
для русского и турецкого языков: https://fastText.cc/docs/en/crawl-vectors.html.

[5]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

шум (%)	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.715	0.593	0.573	0.569	0.564	0.565	0.557
fastText	**0.720**	0.606	0.594	0.581	0.587	0.573	0.569
fastText+speller	0.720	0.638	0.598	0.579	0.585	0.563	0.543
C-Word2Vec	0.689	0.639	0.619	0.601	0.563	0.568	0.515
C-fastText	0.714	0.65	0.646	0.609	0.546	0.536	0.573
C-fastText+speller	0.714	0.642	0.664	0.632	0.588	0.516	0.522
RoVe							
SRU	0.707	**0.701**	0.681	0.661	0.641	0.623	0.611
biSRU	0.715	0.691	**0.687**	**0.662**	**0.644**	**0.637**	**0.631**
CNN-1d	0.632	0.620	0.616	0.610	0.606	0.601	0.596
convLSTM	0.628	0.621	0.616	0.609	0.605	0.592	0.587
LSTM	0.598	0.593	0.583	0.541	0.549	0.513	0.518
biLSTM	0.680	0.650	0.643	0.638	0.627	0.594	0.568
stackedLSTM	0.672	0.658	0.637	0.629	0.606	0.595	0.554

Таблица 2 — Результаты для английского языка задачи определения парафраз

есть уровень искусственного шума соответствует естественному шуму, встречающемуся в данных. Во-первых, система на основе двунаправленного SRU дает наилучшие результаты по метрике ROC-AUC практически для всех уровней шума, кроме нулевого, который является неестественно низким. Во-вторых, система однонаправленного SRU также показывает себя хорошо на низком уровне шума и может использоваться в условиях, когда невозможно использовать двунаправленную архитектуру, например, когда данные поступают для обработки системой в режиме реального времени.

Эксперименты на корпусе SNLI

Результаты продемонстрированы в таблице 2. В данном случае снова предложенная система на основе ячейки biSRU показывает лучшие результаты при высоких уровнях шума.

шум (%)	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.624	0.547	0.593	0.589	0.574	0.565	**0.557**
fastText	0.642	0.570	0.563	0.541	0.517	0.524	0.480
fastText+speller	0.642	0.609	0.498	0.549	0.481	0.450	0.482
C-Word2Vec	0.583	0.55	0.57	0.535	0.533	0.532	0.498
C-fastText	0.619	0.59	0.595	0.587	0.585	0.574	0.561
C-fastText+speller	0.619	0.608	0.592	0.591	0.595	0.554	0.552
RoVe							
SRU	0.627	0.606	0.590	0.581	0.568	0.537	0.531
biSRU	**0.651**	**0.640**	**0.621**	**0.612**	**0.598**	**0.586**	0.536
CNN-1d	0.573	0.569	0.521	0.534	0.517	0.506	0.501
convLSTM	0.568	0.551	0.536	0.529	0.505	0.512	0.487
lstm	0.568	0.553	0.543	0.541	0.526	0.503	0.476
biLSTM	0.619	0.603	0.572	0.561	0.527	0.504	0.498
stackedLSTM	0.617	0.608	0.590	0.529	0.516	0.511	0.505

Таблица 3 — Результаты для задачи определения логической связи (для английского языка).

Эксперименты на корпусе Stanford Sentiment TreeBank

По таблице 1 можно сделать следующие выводы. Результаты системы CNN-1d аналогичны случайному классификатору в этой задаче. Система biSRU имеет наиболее стабильные результаты. Стоит отметить, что качество системы fastText является лучшим на некоторых исследованных шума при использовании проверки орфографии. На других уровнях шума лучше всего себя показывает система Word2Vec. На корпусе SNLI по всей видимости очень важно обучение на конкретных данных, так что классические системы оказываются лучше предлагаемой в этом диссертационном исследовании.

шум (%)	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.649	0.607	0.611	0.583	0.554	0.513	0.457
fastText	0.662	0.639	0.615	0.557	0.524	0.521	0.523
fastText+speller	0.645	0.643	0.573	0.549	0.521	0.448	0.531
C-Word2Vec	**0.721**	0.691	0.677	0.617	0.606	0.610	0.545
C-fastText	0.718	0.704	0.664	**0.659**	0.632	0.621	**0.575**
C-fastText+speller	0.718	**0.708**	**0.698**	0.630	**0.643**	**0.648**	0.549
RoVe							
SRU	0.627	0.606	0.590	0.581	0.568	0.537	0.531
biSRU	0.656	0.640	0.621	0.612	0.598	0.586	0.536
CNN-1d	0.533	0.529	0.511	0.544	0.497	0.509	0.483
convLSTM	0.625	0.591	0.546	0.519	0.515	0.503	0.481
lstm	0.598	0.593	0.553	0.512	0.514	0.499	0.456
biLSTM	0.611	0.607	0.542	0.523	0.528	0.514	0.501
stackedLSTM	0.621	0.598	0.593	0.569	0.586	0.544	**0.563**

Таблица 4 — Результаты для задачи анализа тональности для английского языка.

2.7.2 Эксперименты для русского языка

Ниже приведены результаты систем на русском языке. Сравнение проводится с двумя основными алгоритмами для создания векторных представлений, а именно Word2Vec & fast text. Для системы fastText проведены дополнительные эксперименты с проверкой орфографии.

Для Word2Vec базовой принята система от проекта RusVectores[6], описанная в работте А. Кутузова [93]. Дополнительная использованная система Word2Vec была обучена на Национальном Корпусе Русского Языка (НКРЯ) [7], впервые описанном в Андрющенко [92] и с тех пор значительно дополненном.

[6] http://rusvectores.org/

[7] http://www.ruscorpora.ru/

Также для этой системы использовалась лемматизация mystem [8], описанная в работе Сегаловича [94]. Система fastText используется доступная на GitHub[9].

шум (%)	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.649	0.592	0.581	0.554	0.541	0.532	0.527
fastText	**0.724**	0.709	0.686	0.680	0.608	0.556	0.532
fastText+speller	**0.724**	0.710	0.704	**0.691**	0.623	0.534	0.545
C-Word2Vec	0.704	0.676	0.6	0.546	0.46	0.498	0.450
C-fastText	0.685	0.622	0.598	0.505	0.544	0.523	0.515
C-fastText+speller	0.685	0.649	0.617	0.531	0.524	0.518	0.535
RoVe							
SRU	0.713	0.708	0.701	0.675	0.653	0.602	0.601
biSRU	0.722	**0.714**	**0.708**	0.687	**0.654**	**0.608**	**0.598**
CNN-1d	0.564	0.544	0.538	0.521	0.519	0.498	0.528
convLSTM	0.562	0.554	0.519	0.526	0.509	0.508	0.501
lstm	0.627	0.601	0.548	0.566	0.508	0.512	0.523
biLSTM	0.611	0.607	0.542	0.523	0.528	0.514	0.501
stackedLSTM	0.698	0.659	0.634	0.625	0.608	0.598	0.567

Таблица 5 — Результаты для задачи анализа тональности для русского языка.

Предлагаемая в этом диссертационном исследовании система обладает близким к лучшему или лучшим качеством на всех исследованных уровнях шума (см. таблицу 5). В экспериментах на этом корпусе оказалось более важным предобучение на система большом корпусе данных. Можно предположить, что это позволяет выучить семантические свойства слов, такие как присущая им тональность.

Стоит отметить, что уровень шума важен для русского языка больше, чем для английского языка, так как качество всех протестированных систем падает быстрее с ростом уровня шума.

На задаче определения парафраз для русского языка можно наблюдать схожую картину (см. таблицу 6) - предлагаемая система показывает лучшее

[8] https://tech.yandex.ru/system/

[9] https://github.com/facebook_research/fastText

шум (%)	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.800	0.727	0.694	0.618	0.607	0.527	0.511
fastText	**0.860**	0.788	0.728	0.632	0.596	0.589	0.523
fastText+speller	**0.860**	0.817	0.791	0.726	0.609	0.599	0.578
C-Word2Vec	0.726	0.631	0.666	0.581	0.596	0.582	0.592
C-fastText	0.770	0.766	0.732	0.642	0.648	0.694	0.599
C-fastText+speller	0.770	0.75	0.749	0.668	0.656	0.678	0.532
RoVe							
SRU	0.827	**0.844**	0.803	0.781	0.643	0.625	**0.611**
biSRU	0.843	0.823	**0.818**	**0.789**	**0.698**	**0.647**	0.596
CNN-1d	0.612	0.601	0.533	0.519	0.523	0.52	0.511
convLSTM	0.632	0.598	0.551	0.534	0.542	0.501	0.509
lstm	0.697	0.649	0.634	0.602	0.587	0.545	0.512
biLSTM	0.701	0.682	0.646	0.623	0.619	0.591	0.578
stackedLSTM	0.723	0.687	0.641	0.631	0.632	0.599	0.583

Таблица 6 — Результаты для задачи определения парафраз для русского языка.

качество, то есть большую устойчивость к шуму на всех уровнях шума, кроме нулевого. Среди рассматриваемых вариантов системы лучшую устойчивость показывают два варианта, основанные на использовании модуля обработки контекста с ячейками SRU и biSRU.

2.7.3 Эксперименты для турецкого языка

Как упоминалось выше, эта система подходит морфологически богатых языков, таких, как турецкий. Для проверки используется только один набор данных. В качестве базовой используется система Word2Vec, обученная на корпусе ‘42 bin haber’ (42 тыс. новостей), описанном в работе Йылдырым [95]. Для fastText системы используется система с официального сайта для турецкого языка. На этапе тестирования усредняются все вектора слов, известных систе-

ме. Предварительно происходит стемминг с помощью стеммера Snowball [96] для fastText и Word2Vec.

шум	0	5	10	15	20	25	30
Базовые системы							
Word2Vec	0.647	0.587	0.586	0.572	0.534	0.513	0.475
fastText	0.632	0.609	0.595	0.537	0.514	0.520	0.503
RoVe							
SRU	0.647	0.632	0.602	0.591	0.568	0.532	0.517
biSRU	**0.718**	**0.679**	**0.641**	**0.604**	**0.587**	**0.583**	**0.556**
CNN-1d	0.513	0.518	0.521	0.504	0.499	0.519	0.503
convLSTM	0.511	0.491	0.503	0.499	0.507	0.523	0.513
lstm	0.568	0.567	0.543	0.531	0.524	0.518	0.506
biLSTM	0.604	0.591	0.581	0.562	0.548	0.532	0.531
stackedLSTM	0.601	0.596	0.584	0.579	0.536	0.524	0.521

Таблица 7 — Результаты для задачи поиска парафраз для турецкого языка

Результаты представлены в таблице 7. Как видно из таблицы, главная особенность устойчивости шуму предлагаемой системы заметна и в этом эксперименте. Система на основе ячейки biSRU показывает результаты лучше систем, модуль обработки контекста которых основан на других ячейках и принципах.

2.8 Изучение влияния шума

Все результаты, о которых говорилось выше, были протестированы на наборах данных с тремя типами шума (перестановка, вставка и удаление букв), применяемыми одновременно. Система по определению инвариантна к буквенным перестановкам, поэтому не включили этот тип шума в отдельные эксперименты. Однако перестановка не изменяет векторное представление слова только тогда, когда она происходит за пределами сегментов B и E слова, в противном случае встраивание изменяется как B и E сохранить порядок букв. Поэтому будет сравниваться эффект случайных буквенных перестановок соседних букв.

Дополнительно проведено сравнение четырех видов шума:

– только вставка букв,

– только удаление,

– вставка и удаление,

– только перестановки.

Как видно из приведенных экспериментов эффект перестановки зависит от языка и набора данных. Это ухудшает оценки сильнее для языков с более короткими словами 4, потому что эти перестановки часто происходят в сегментах B и E слов. В проведенных экспериментах по определению парафраз и логической связи все четыре типа шума произвели одинаковый эффект на наборы данных на английском языке, где средняя длина слов составляет от 4 до 4,7 символов. С другой стороны, русскиоязыные и туркоязычный наборы данных (средняя длина слова – 5,7 символов соответственно) были более устойчивы к буквенным перестановкам, чем к другим типам шума.

Однако, это справедливо только для задач, где результат вычисляется как косинусная мера сходства между векторами, т. е. векторами полностью определяется качество предсказания. В задаче анализа тональности, где обучался наивный байесовский классификатор, все типы шума оказывали сходное влияние на качество системы как для английского, так и для русского.

2.9 Анализ результатов сравнения систем на прикладных задачах обработки текстов

Как можно заметить, ни одна из систем не устойчива к опечаткам на 100% - их качество падает, когда добавляется шум. Тем не менее, это снижение намного больше для базовых систем, что означает, что RoVe менее чувствителен к опечаткам. Рисунок 2.6 показывает, что, хотя все системы отражают один и тот же результат на чистых данных (не содержащих опечаток, индуцированных смоделированным шумом), RoVe превосходит базовые системы по мере повышения уровня шума. Из всех вариантов модуля обработки контекста для системы RoVe вариант на основе ячейки двунаправленный SRU дает лучший результат, незначительно превосходя алаог, основанный на ячейке SRU. Стоит отметить, что разница между biSRU и SRU версиями модуля обработки

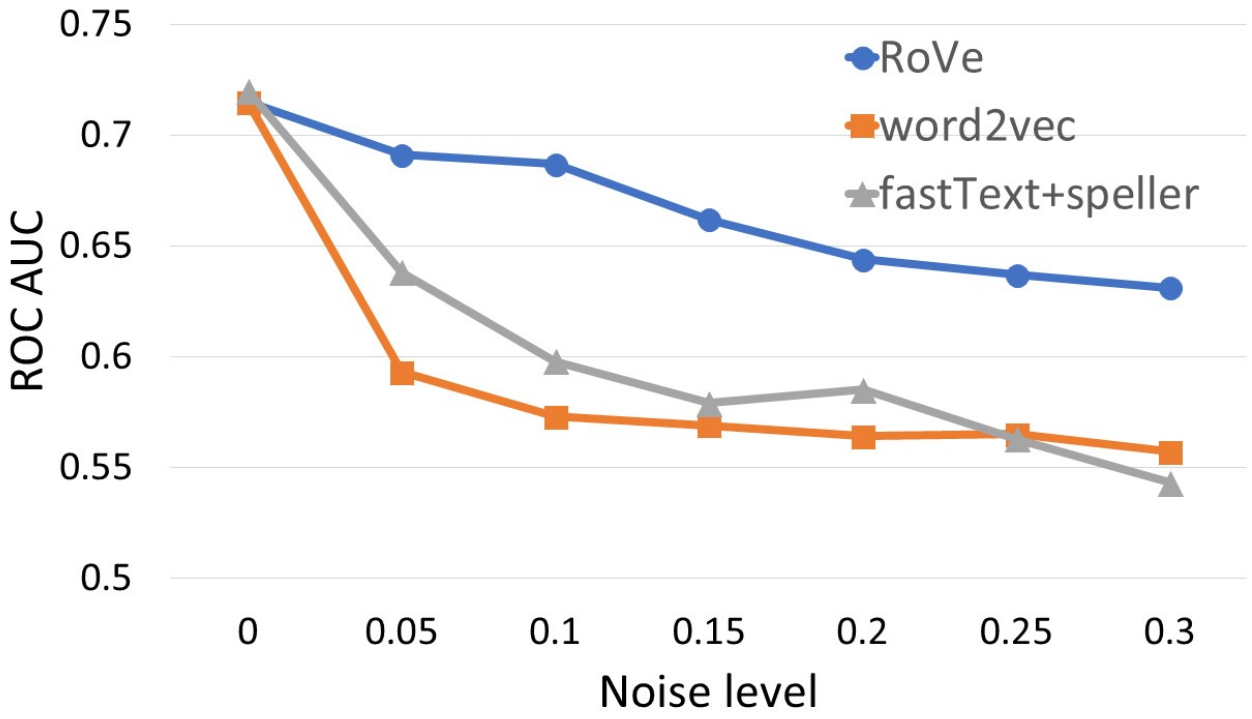

Рисунок 2.6 — Сравнение систем RoVe, Word2Vec и fastText на текстах с увеличиваемым уровнем шума на задаче обнаружения парафраз для английского языка.

контекста в системе RoVe больше для русских и турецкого наборов данных. Это можно объяснить свободным порядком слов в русском языке и структуре предложений “субъект-объект-глагол” в турецком языке (в отличие от “субъект-объект-глагол” в английском языке).

Интересно, что использование систем исправления опечаток не гарантирует улучшения: система fastText + spell-checker не всегда превосходит обычный fastText, а ее оценка нестабильна (см. результаты для русского, где fastText + spell-checker лучше, чем fastText на 10% шума и резко падает на 20%).

Это может быть объяснено тем, что система проверки орфографии сама делает ошибки, например, иногда она может изменить правильное слово на неправильное. Или увидев какую-то незначительную ошибку в данных увеличить ее влияние, изменив слово на правильное, но имеющее абсолютно другой смысл.

Описываемая система имеет две ключевые невзаимосвязанные особенности: обработка слов не из словаря и контекстная зависимость векторных представлений. Чтобы увидеть, насколько каждая из них способствует конечному качеству, они были протестированы отдельно.

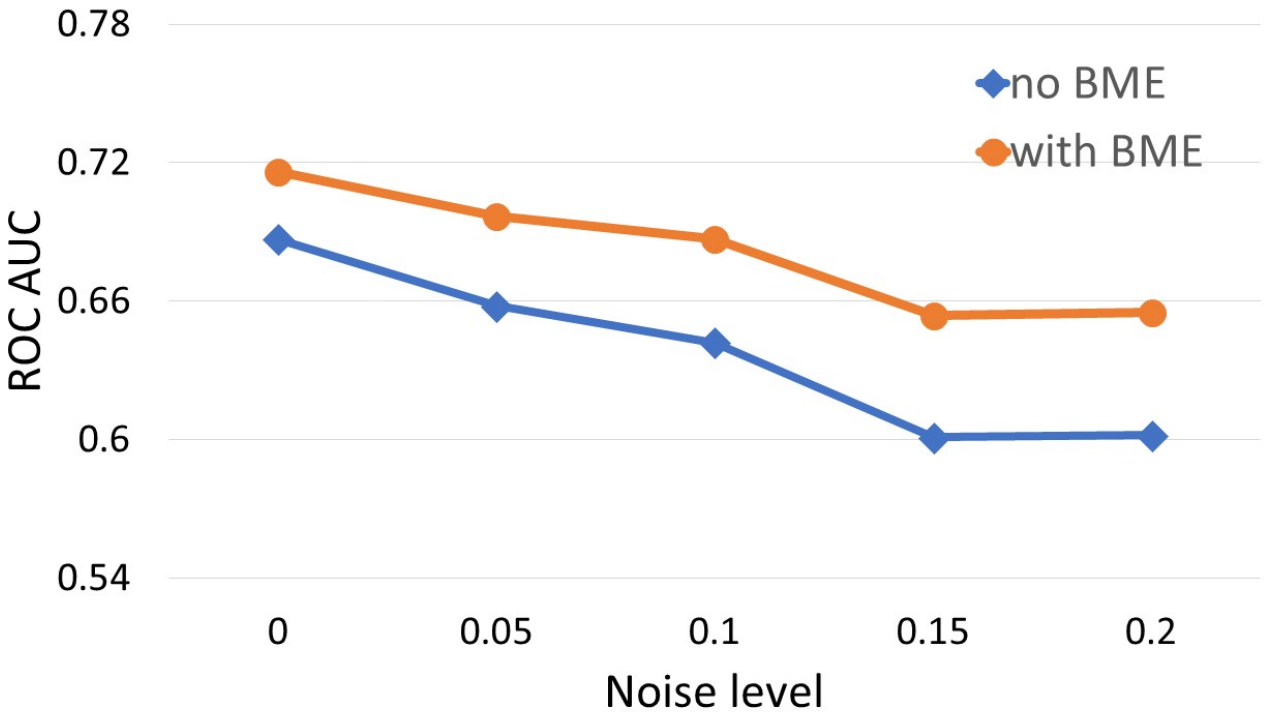

Рисунок 2.7 — RoVe с и без BME (paraphrase detection task for English).

Для того, чтобы проверить важность представления BME, это представление было исключено из системы, т.е. слово рассматривается как мешок букв (кодируется только сегмент M из BME). Таким образом, система все еще имеет открытый словарь, но менее выразительна. На рисунке 2.7 показано качество систем с представлением BME и без оного в задаче определения парафраз для английского языка.

Можно заметить, что представление BME не делает систему абсолютно устойчивой к опечаткам – для обоих случаев оценки качества уменьшаются до одного уровня на высоких уровнях шума. Однако BME повышает качество векторов для всех уровня шума. Можно считать доказанной гипотезу, что префиксы и суффиксы содержат много информации, которую не следует отбрасывать. Результаты для других языков и задач показывают ту же тенденцию.

В следующем эксперименте отбрасывается контекстную зависимость векторов слов. Кодировщик заменяется проекционным полносвязным слоем, который преобразует представление слова BME в 300-мерный вектор.

На рисунке 2.8 показана производительность этой системы при решении задачи определения парафраз для английского языка. Качество близко к случайному (случайный классификатор имеет ROC AUC 0.5). Кроме того, это не согласуется с уровнем шума - в отличие от ранее приведенных результатов,

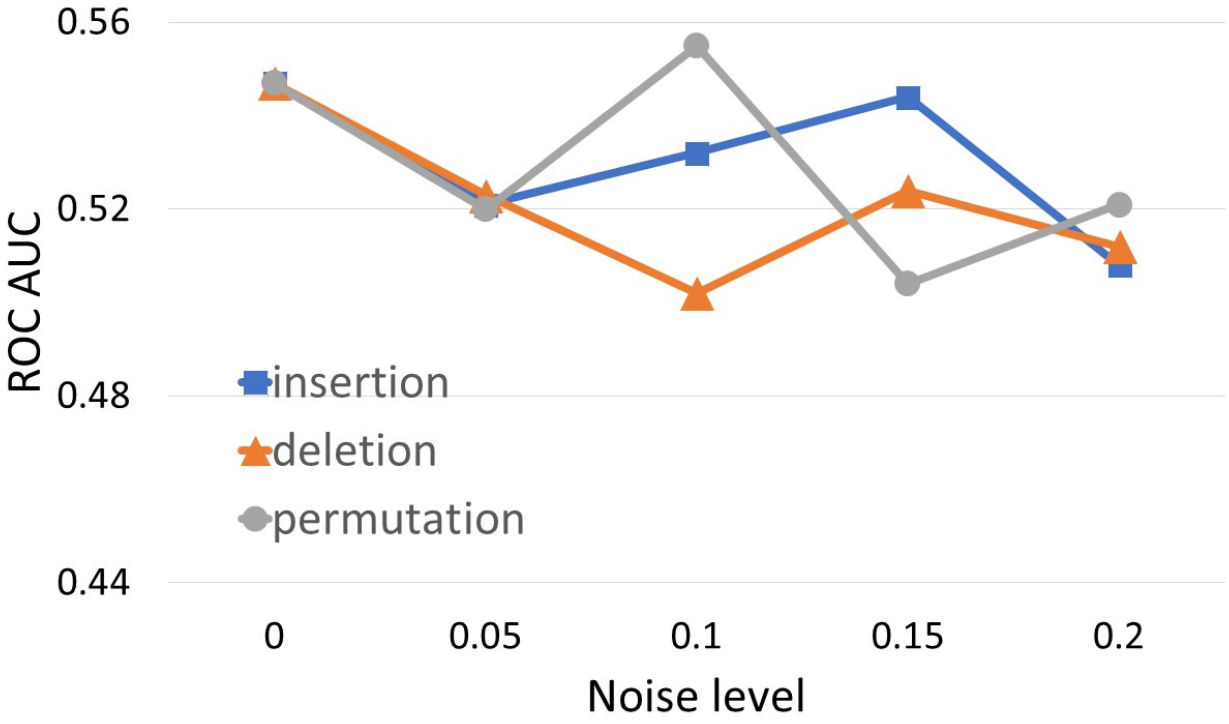

Рисунок 2.8 — RoVe без контекстной информации (обнаружение парафраз для английского языка).

качество не уменьшается монотонно с увеличением уровня. Результат этого эксперимента предсказуем, если учесть, что кодировщик является единственной обучаемой частью системы, таким образом, это та часть, которая в основном отвечает за качество векторных представлений слов.

2.10 Выводы к главе 2

В данной работе представлена RoVe – новая система для обучения векторных представлений слов, которая устойчива к опечаткам, которые сохраняют содержимое контекста. Данная система реализует описываемый метод построения устойчивых к шуму программных систем. Система содержит две ключевых части: модуль обработки контекста и модуль обработки слов не из словаря. По результатам экспериментов можно говорить, что каждая из частей системы - BME и кодировщик контекста - вносит важный вклад в эффективную работу системы.

В отличие от других подходов, этот метод не задает явного словаря. Векторное представление слова формируется из векторных представлений его символов, поэтому RoVe может генерировать векторное представление для любой строки. Это уменьшает влияние орфографических ошибок, слова с замененными символами имеют те же представления, что и их исходные версии, и слова с пропущенными или дополнительными символами имеют вектора, близкие к одной из их исходных словоформ.

Также в данной главе было продемонстрировано, что метод сравнения систем позволяет сделать выводы об их устойчивости к различных уровням шума.

Система RoVe была проверена с различными кодировщиками и было выявлено, что ячейка SRU (Simple Recurrent Unit) наилучшим образом из всех протестированных подходит для проверенных задач, особенно в варианте двунаправленного SRU. Проведенные эксперименты показали, что предложенная система более устойчива к опечаткам, чем системы Word2Vec и fastText, обычно используемые для обучения векторных представлений слов. Их качество существенно падает при добавлении шума даже небольшого уровня. Кроме того было проведено исследование в результате которого было показано, что система RoVe более устойчива к опечаткам, чем система fastText вместе с системой для исправления ошибок в текстах (по всей видимости, из-за ошибок, которые допускает система проверки орфографии).

RoVe может производить осмысленные векторные представления для слов и словоформ в морфологически богатых языках - русском и турецком. Можно предположить, что это свойство будет выполнено и для других морфологически богатых языком, однако это не проверялось. В качестве направления будущих исследований можно назвать проверку RoVe на других языках. Это потребует настройки длин сегментов B и E для префиксов и суффиксов соответственно.

Глава 3. Устойчивая к шуму классификация текстов

Множество приложений для классификации текста, таких как анализ тональности или распознавание намерений, связано с данными, создаваемыми пользователем, где не может быть гарантирована правильная орфография или грамматика.

Классический подход к векторизации текста, такой как кодирование слов one-hot или TF-IDF-кодирование, сталкивается с проблемой отсутствия слов в словаре, учитывая огромное разнообразие орфографических ошибок. Существуют работы, демонстрирующие для задач с небольшим шумом на общеупотребимых наборах данных [97; 98], не все системы ведут себя хорошо с данными реального мира, такими как комментарии или твиты.

Эта глава организована следующим образом. В разделе 3.1 описываются системы, о которых идет речь в этой главе. Разделы 3.2 и 3.3 описывают постановку эксперимента и полученные результаты с последующей интерпретацией соответственно.

3.1 Системы классификации текстов

Метод построения систем, устойчивых к шуму, в задаче классификации текстов

Система классификации текстов может быть представлена, как совокупность модулей векторного представления слов и принятия решения о присвоении метки класса и процедуры обучения программной системы. *Метод построения устойчивых к шуму программных систем* для данной задачи состоит из следующих этапов:

– модификации модуля векторного представления слов,

– модификации процедуры обучения программной системы.

Метод построения устойчивых к шуму программных систем для данной задачи требует разделения программной системы на несколько компонент:

а) компоненту векторного представления слов, которая может разделяться на компоненту буквенного представления слова и представления слова, как целого; б) компоненту принятия решения о присвоении метки класса; в) описания обработки входных данных и передачи данных между компонентами системы.

Экспериментально была проверена производительность следующих систем.

3.1.1 CharCNN

Текст представлен в виде последовательности символов, изначально она была предложена в работе Кима [99]. Эта система состоит из слоя векторного представления символов, слоя свертки с 256 фильтрами; размер ядра равен 15, а шаг - 2, за ним следует слой субдискретизации с размером ядра 64 и шагом 32, используется функция максимума. Данная последовательность слоев рассматривается в качестве компоненты векторного представления слов. Выход данной компоненты передается компоненте принятия решения. Данная компонента состоит из процедуры дропаут и преобразование 256-мерного скрытого вектора к 2 измерениям с помощью полносвязного слоя. Архитектура представлена на рис. 3.1.

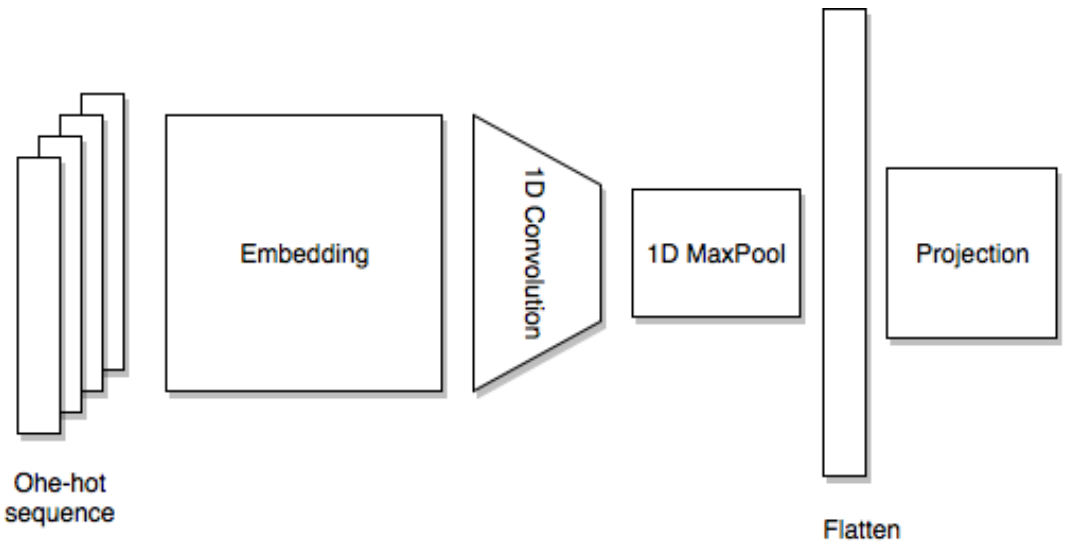

Рисунок 3.1 — CharCNN

3.1.2 FastText

Текст представлен как последовательность векторов из программной системы FastText, выступающей в качестве компоненты векторного представления слов. Последовательность векторов подается на вход программе представляющей собой нейронную сеть, содержащую слой GRU размером 256 ячеек. Далее применяется процедура дропаут к последнему скрытому состоянию слоя GRU, а полученный вектор проецируется в 2-мерное пространство. Данная программа рассматривается, как компонента приятия решения данной системы.

3.1.3 CharCNN-WordRNN

Данная программная система повторяет программную систему, описанную в работе Кима [60]. Каждое слово представляется в виде последовательности символов, а текст представляется как последовательность представлений слов, что рассматривается в качестве компоненты векторного представления слов. Слова получают векторное представление с помощью сверточного слоя с размером ядра 5 и субдискретизирующего слоя с максимумом по времени. Векторные представления поступают в программный модуль нейронной сети, содержащий слой GRU размерности 128 ячеек, слой, реализующий процедуру дропаут и слой проецирования - аналогично предыдущему разделу. Данный программный модуль рассматривается, как компонента приятия решения данной системы.

3.1.4 RoVe

Эта система аналогична системе FastText, за исключением того, что используются вектора, порождаемые системой RoVe вместо векторов, порождаемых системой fastText.

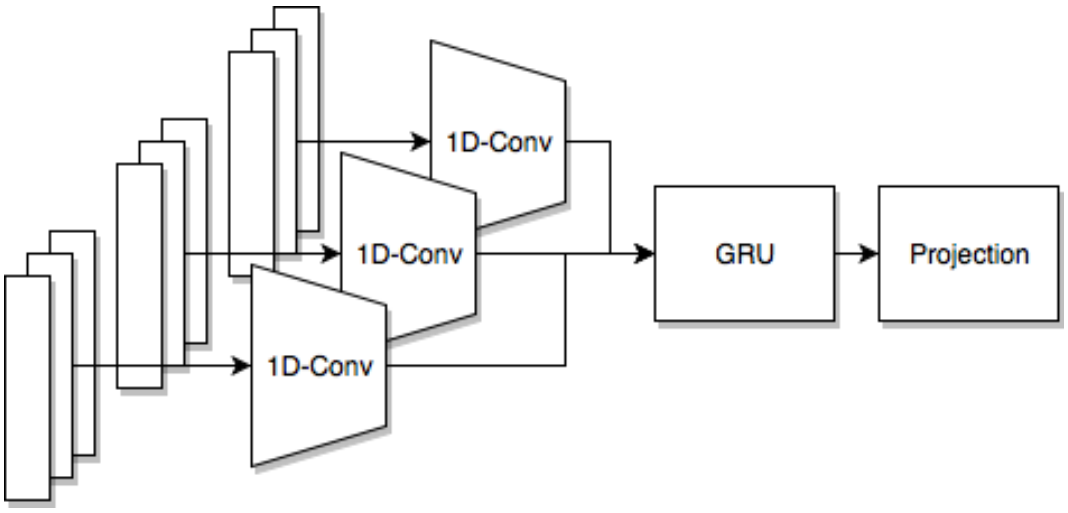

Рисунок 3.2 — CharCNN-WordRNN

3.2 Эксперименты по сравнению систем классификации текстов

3.2.1 Наборы данных для задачи классификации текстов

Были проведены эксперименты на двух наборах данных: набор данных с отзывами об авиакомпаниях на английском языке и набор отзывов об автомобилях и ресторанах для русского языка.

Набор данных Airline Twitter Sentiment состоит из 14.485 твитов, описывающих впечатления пользователей от полета определенными авиакомпаниями. сообщения из набора размечены по трем классам – позитивный, негативный и нейтральный. Набор данных является публично доступным.[1]

Набор данных SentiRuEval-2015 был описан в работе Лукашевич [89]. Набор данных состоит двух доменов т.н. “ресторанного” и “автомобильного”, для каждого из доменов. Автомобильный домен содержит отзывы о марках автомобилей. Автомобильный домен содержит 207 и 205 сообщений в обучающей и тестовой выборках соответственно. Ресторанный домен содержит отзывы о ресторанах. Ресторанный домен содержит 207 и 205 сообщений в обучающей и тестовой выборках соответственно. Все сообщения из набора размечены по четырем классам: содержащие позитивный отзыв, негативный отзыв, нейтраль-

[1]Доступен для скачивания по ссылке: https://www.kaggle.com/crowdflower/twitter-airline-sentiment.

ный отзыв, или отзыв, содержащий и позитивную и негативную тональность одновременно.

3.2.2 Метод сравнения систем классификации текстов

Метод сравнения программных систем на предмет устойчивости к шуму в задаче классификации текстов имеет следующий вид. Следует провести три вида экспериментов, с наложением и отсутствием наложения шума на входные данные тестируемых систем с разными уровнями шума.

Система, тестируемая по предлагаемому методу, обучается на незашумленном либо зашумленном тексте, далее обученной системе подается на вход искусственно или естественно зашумлённый текст. В случае искусственно зашумленного текста уровень шума в тексте заранее известен и составлял от 0% до 20%. Согласно работе Кучержана и Брилла [2], реальный уровень шума в пользовательских текстах составляет 10-15%.

Шум моделируется следующим образом:

– вероятность вставки буквы после текущей,

– вероятность того, что буква будет пропущена,

для каждой буквы входного алфавита для каждой задачи. На каждой входной строке выполняются случайные вставки и удаления букв с помощью смоделированных вероятностей. Оба типа шума добавляются одновременно.

Чтобы продемонстрировать устойчивость к шуму, была выполнена проверка орфографии [2] описанных наборов данных и искусственно введен в них шум.

Были проведены три типа экспериментов:

– данные обрабатываются системой проверки орфографии, а после в них добавляется описанный искусственный шум; описанная процедура применяется как тренировочной, так и к тестовой выборке;

– берутся исходные, неизмененные данные;

– данные обрабатываются системой проверки орфографии, а после в них добавляется описанный искусственный шум; описанная процедура при-

[2]Для правки орфографии использоваться общедоступный промышленный механизм проверки орфографии Yandex.Speller.

Система \ Набор данных	SentiRuEval-2015	Airline Twitter Sentiment
CharCNN	0.40	0.77
FastTextGRU	0.45	0.76
CharCNN-WordRNN	0.39	0.81
RoVe	0.38	0.78

Таблица 8 — Результаты экспериментов на неизменных наборах данных. F_1 на тестовой выборке.

меняется только к тренировочной выборке, тестовая выборка остается без изменений.

Эти эксперименты призваны продемонстрировать устойчивость тестируемых систем для искусственного и естественного шума. Под устойчивостью автор понимает степень снижения качества системы с введением шумов различного уровня вероятности. Таким образом более устойчивой полагается та система, качество которой снижается меньше с ростом уровня шума. В качестве меры качества выбрана мера F_1 для классификации.

3.3 Результаты экспериментов для задачи классификации текстов

Все системы обучаются с размером батча 32, оптимизатором Adam и дропаутом для последнего полносвязного слоя с вероятностью 0,5. Функция потерь - перекрестная энтропия. CNN инициализируются с помощью инициализации Х. Глоро [100] с нормальным распределением.

Ниже представлены результаты обучения и тестирования систем без проверки орфографии или искусственного шума.

Результаты в таблице 8 представлены для сравнения, чтобы сравнить со следующими результатами в шумной среде.

3.3.1 Набор данных SentiRuEval-2015

На рис. 3.3 представлены результаты экспериментов в следующих условиях: обучающая выборка очищается от естественного шума, далее в нее вводится искусственный; тестовая выборка также подвергается описанной процедуре. На графике видно, что система FastText показывает лучшее качество практически для всех уровней шума. Система RoVe в этом эксперименте показывает второй результат для всех уровней шума, кроме самых высоких, где она выходит оказывается более устойчивой.

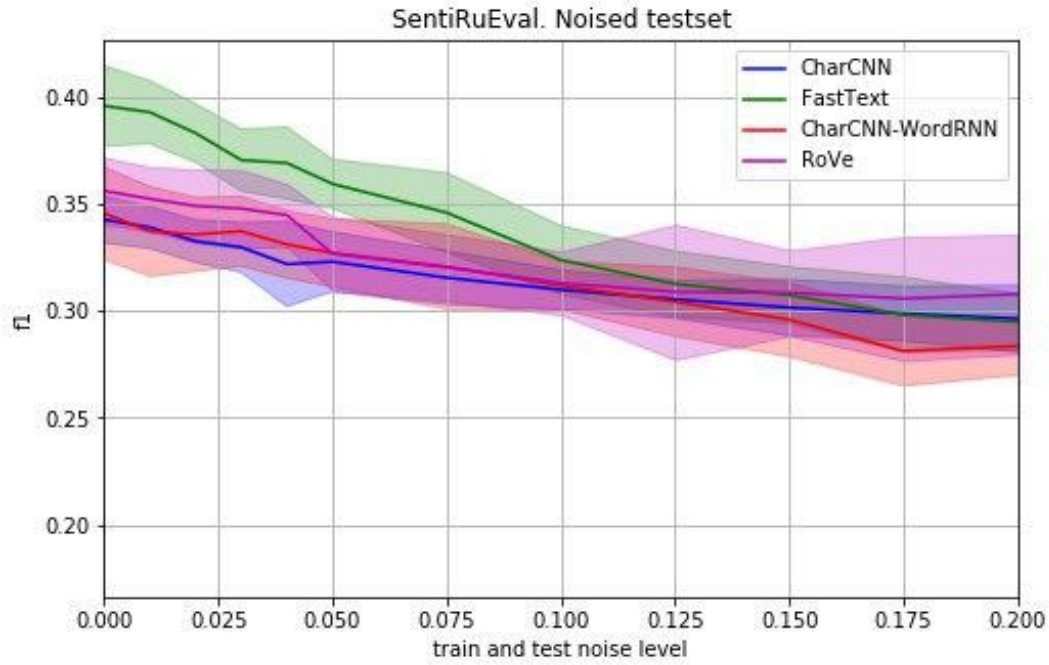

Рисунок 3.3 — Набор данных SentiRuEval-2015. Обучение на данных с исправленными естественными опечатками и добавленными искусственными, проверка на тестовых данных с исправленными естественными опечатками и добавленными искусственными с тем же уровнем шума, что и на обучающей выборке.

На рис. 3.4 представлены результаты в следующих условиях: обучающая выборка очищается от естественного шума и далее в нее вводится искусственный; выстовая выборка остается неизмененной. Все системы ведут себя подобным образом, если сравнивать с предыдущим экспериментом. Система FastText показывает лучшее качество на всем тестируемом диапазоне вероятностей шума. На втором месте система RoVe.

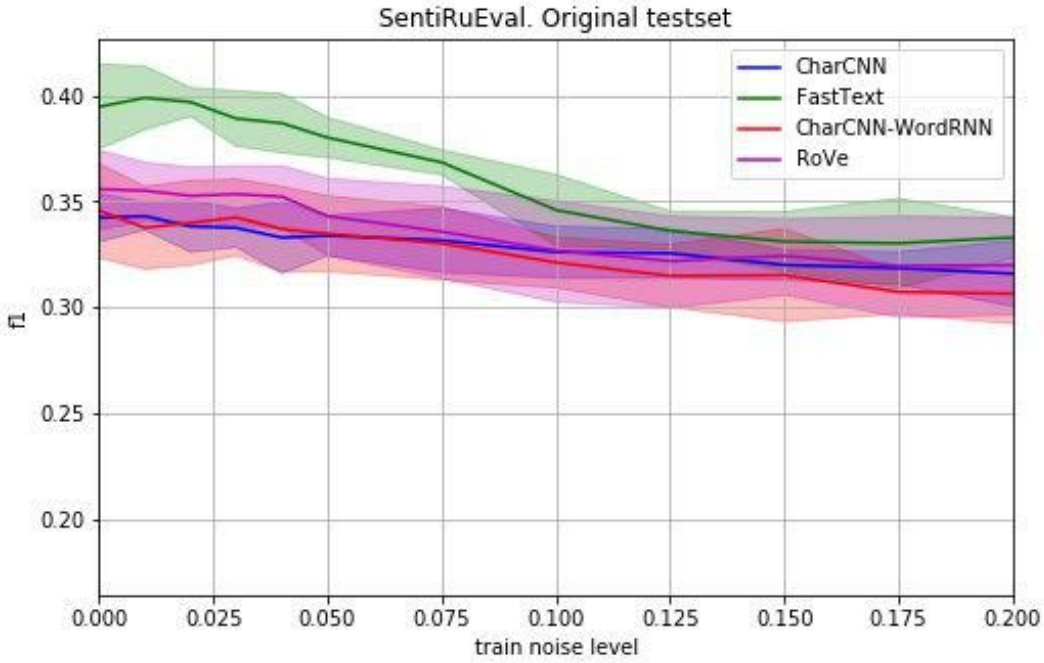

Рисунок 3.4 — Набор данных SentiRuEval-2015. Обучение на данных с исправленными естественными опечатками и добавленными искусственными, проверка на данных с исправленными естественными опечатками и добавленными искусственными с тем же уровнем шума, что и на обучающей выборке.

3.3.2 Набор данных Airline Twitter Sentiment

На рис. 3.5 представлены результаты в следующих условиях: обучающая выборка очищается от естественного шума и вводится искусственный; тестовая выборка также подвергается описанной процедуре. Можно заметить, что система FastText показывает лучшее качество на низких уровнях шума. Система RoVe в этом эксперименте показывает второй результат для низких уровней шума, а на высоких показывает лучший результат в силу заложенного в нее свойства устойчивости.

На рис. 3.6 представлены результаты в следующем окружении: обучающая выборка очищается от естественного шума и вводится искусственный; набор данных для тестирования остается неизменным. В этом эксперименте системы демонтируют поведение подобное демонстрируемому в прошлом эксперименте. Система FastText показывает лучшее качество практически для всех уровней шума, кроме области больших шумов. Система RoVe в этом эксперименте по-

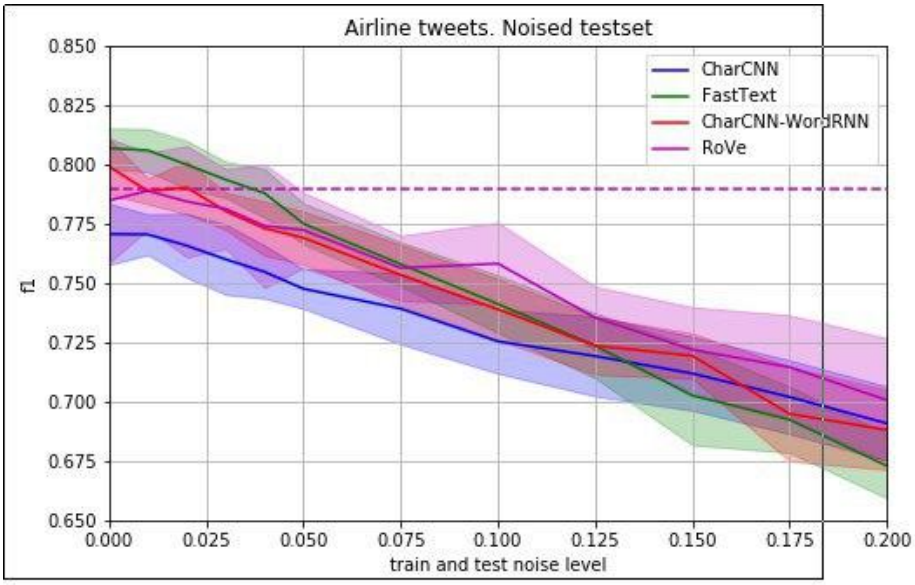

Рисунок 3.5 — Набор данных Airline Twitter Sentiment. Системы обучались на данных с исправленными естественными опечатками и добавленными искусственными и проверялись на исходных данных. F_1 на тестовой выборке.

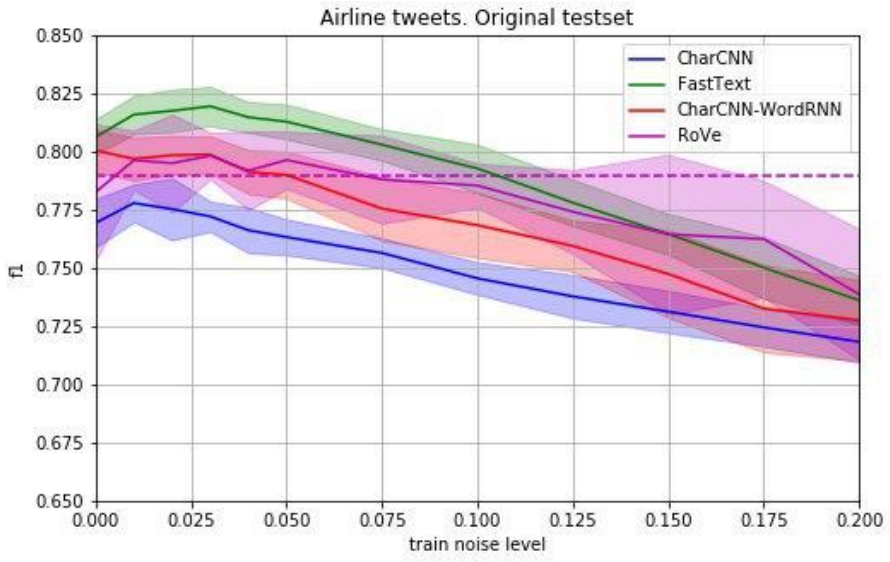

Рисунок 3.6 — Набор данных Airline Twitter Sentiment. Тренировка на данных с исправленными естественными опечатками и добавленными искусственными, тест на данных с исправленными естественными опечатками и добавленными искусственными с тем же уровнем шума, что и на тренировочной выборке.

казывает второй результат для всех уровней шума, кроме самых высоких, где она выходит оказывается более устойчивой.

3.4 Выводы к главе 3

Проведенные эксперименты позволяют оценить устойчивость к шуму распространенных современных систем для классификации текста. Продемонстрировано, что некоторые из популярных систем являются более устойчивыми, но в области больших шумов наиболее устойчивой оказалась система RoVe, обладающая изначально заложенным свойством устойчивости к шуму в силу построения по предлагаемому методу построения устойчивых к шуму систем.

Стоит также отметить, что результаты на корпусе SentiRuEval в среднем существенно ниже (оставляя за скобками разницу в языках), так как эти данные содержат 4 класса, а не три, как в наборе данных Airline Tweets.

Автор рассматривает будущие улучшения и расширения этой работы в трех основных направлениях: постановка экспериментов с другими видами шума, тестирование не освещенных в этой работе архитектур для классификации текста и эксперименты с другими наборами данных.

Глава 4. Распознавание именованных сущностей в шумных текстах

В последние годы большое внимание уделялось области распознавания именованных сущностей (named entity recognition, NER). Ранее технология NER была в основном востребована компаниями, для которых было важны упоминания о них в традиционных СМИ, таких как газеты. Эти медиа-источники имеют проверку орфографии и коррекцию, поэтому могут считаться “чистыми” в терминах данной диссертационной работы. Но в настоящее время социальные медиа стали расти и привлекать все больше внимания со стороны потребителей технологии NER. Эта область имеет свои особенности по сравнению с традиционными СМИ, такие как использование специфического языка (т. е. сленга) и многочисленные опечатки и грамматические ошибки. Специфический язык может быть обработан с помощью современных векторных представлений слов, имеющих семантическое сходство, например, система Word2Vec, описанная в работе Т. Миколова [6]. Но проблема опечаток требует отдельного рассмотрения. Современные системы проверки орфографии не решают эту проблему полностью, показывая качество до 85% [4].

Существующие лучшие на сегодняшний день решения для русского языка, представленная в работе Аня [101], и английского, представленная в работе Лампля [102], а также французского, описанной в работе автора [12], языков, построены по одному принципу. В качестве входа для системы используются векторные представления слов, обогащенные векторными представлениями, полученными на основе побуквенного представления слова. Описанные векторные представления слов поступают на модуль обработки контекста LSTM и уже выход этого модуля поступает для обработки на модуль CRF для построения финальной последовательности меток. Можно заметить, что в описанной архитектуре не предусмотрено явной устойчивости к опечаткам. Таким образом, нам нужно уделять больше внимания устойчивости систем распознавания именованных сущностей к этому типу шума.

В этой главе представлены следующие результаты:

- результаты устойчивости к шуму для английского и русского языков на новостных корпусах;
- результаты устойчивости к шуму для французского языка на корпусе социальных сетей;

– результаты для французского корпуса без искусственного шума, превосходящие описанные в литературе на сегодняшний день.

4.1 Базовая архитектура системы распознавания именованных сущностей

Архитектура базовой системы следует [101], за исключением использования различных векторных представлений слов, которые будут описаны ниже.

Для каждого слова делается два представления: на уровне слова и на уровне символов. Представления на уровне слов происходят из предварительно обученной системы векторных представлений слов; в экспериментах используются системы Word2Vec, описанной в работе Миколова и соавторов [6], и fastText, описанной в работе Миколова и соавторов [97], а также случайные векторные представления и векторные представления, выучиваемые в процессе обучения системы.

Вторая часть векторного представления для слова генерируется сверточной нейронной сетью, которая которая работает с побуквенным представлением слова и для каждого символа использует его векторное представление, выученное в процессе обучения системы.

Эти две части конкатенируются и проецируются полносвязным слоем для получения вектора требуемой размерности. Эти векторы приходят в блок обработки последовательностей (контекста), который может быть рекуррентной нейронной сетью (RNN) или одномерной сверточной нейронной сетью, в которой операция свертки происходит по измерению времени. В проведенных экспериментах в качестве RNN используется biLSTM.

Результирующая векторная последовательность подается в условное случайное поле (CRF) для получения окончательной последовательности тегов. Базовая архитектура системы изображена на фиг. 4.1.

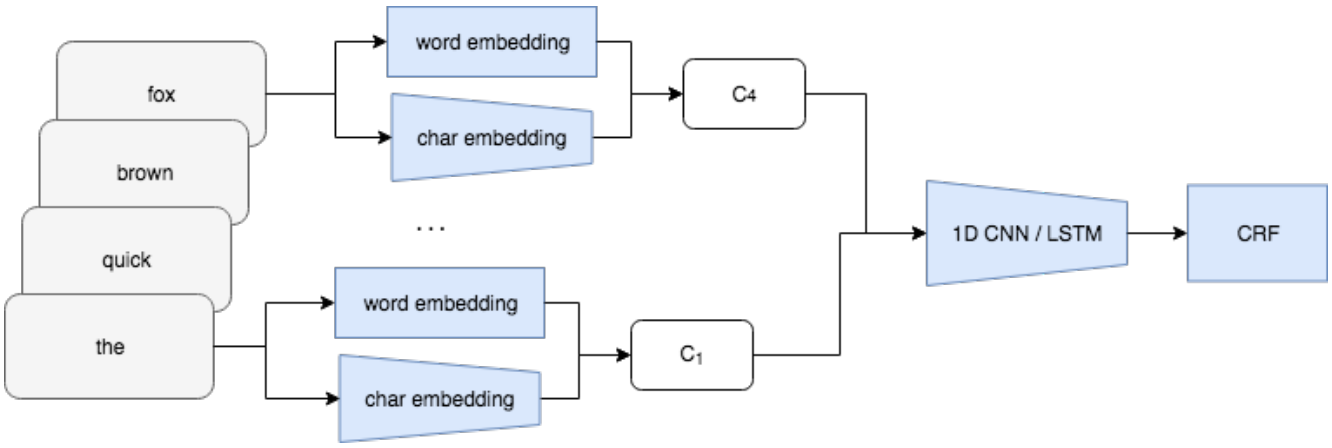

Рисунок 4.1 — Базовая архитектура изучаемых систем для задачи распознавания именованных сущностей - biLSTM-CRF.

4.2 Наборы данных для задачи распознавания именованных сущностей

В экспериментах использовались три набора данных: Persons-1000 для русского языка, CAp'2017 для французского и CoNLL'03 для английского языка.

4.2.1 Набор данных CoNLL'03

Набор данных CoNLL'03 состоит из новостных статей новостного корпуса Reuters 1996, представленном в работе [103]. Набор данных с аннотацией по схеме **BIO** (**B**egin, **I**ntermediate, **O**ther) и содержит четыре типа именованных сущностей: организации (**ORG**anisation), локации (**LOC**ation), персоны (**PER**son) и разное (**MISC**ellaneous).

Использовалась только англоязычная часть этого корпуса, которая содержит 946 документов в обучающей, 216 документов в валидационной и 231 документ в тестовой выборке.

Важно отметить, что именованные сущности из тестовой выборки отличаются от таковых для обучающей и валидационной выборок.

4.2.2 Корпус Persons-1000

Persons-1000 содержит новостные документы из различных российских информационных агентств с аннотированными тегами **PER**, **ORG**, **LOC**, **MEDIA** (для названий СМИ) и **GEOPOLIT** (для названий стран и подобных сущностей). Количество документов в корпусе составляет 1000. Схема разметки и процедура разделения намеренно были выбраны аналогично CoNLL. Коллекция была впервые описана в работе Власова и соавторов [104], в работе Можаровой и Лукашевич [105] была представлена дополнительная разметка для этого корпуса, которая использовалась в экспериментах, описываемых в этой диссертационной работе.

4.2.3 Корпус CAp'2017

Набор данных, описанный в работе [106], состоит из коротких сообщений (твитов), аннотированных 13 типами сущностей. Язык этого корпуса - французский. Количество документов (твитов) в наборе данных составляет 3000 для обучающей выборки и 3685 для тестовой выборки. Корпус также размечен BIO-разметкой аналогично CoNLL.

4.3 Эксперименты с вариантами системы biLSTM-CRF

Для набора данных CAp'2017 была произведена предобработка в виде проверки орфографии, т.к. этот набор данных заведомо содержал опечатки.[1] Как CoNLL'03, так и Persons-1000 полагаются не содержащими орфографических ошибок, поэтому автоматическая коррекция орфографических ошибок не выполнялась.

[1]Использовалась система проверки орфографии, доступная как часть Microsoft Cognitive Toolkit.

Следует также отметить, что никакой другой предобработки этих корпусов не проводилось.

4.3.1 Метод сравнения систем распознавания именованных сущностей

Метод сравнения программных систем на предмет устойчивости к шуму в задаче распознавания именованных сущностей, аналогично главе 3, имеет следующий вид. Необходимо провести три вида экспериментов, с наложением и отсутствием наложения шума на входные данные тестируемых систем с разными уровнями шума.

В качестве метода сравнения систем предлагается метод внесения шума в тестовые данные. Для сравнения системам, обученным на зашумленных либо незашумленных данных подается на вход текст с внесенным шумом и замеряется качество на некоторой задаче.

Для проведения экспериментов в наборы данных искусственно привносится шум. В качестве шума моделируется вероятность замены текущего символа в строке случайным символом из алфавита, выбранным равновероятно, для каждого символа входной строки. Для каждого символа входной строки случайная замены выполняется замена символов с заданной вероятностью. Системы проверяются с различным уровнем шума - от 0% (без шума) до 20%. Согласно работе Кучержана и Брилла [2], реальный уровень шума в создаваемых пользователями текстах составляет 10-15%.

Были проведены три типа экспериментов:

– обучающая и тестовая выборки не изменяются, и искусственный шум не добавляется;
– производится проверка орфографии и тестовой, и обучающей выборок; искусственный шум добавляется в обучающую и тестовую выборки одновременно;
– производится проверка орфографии для обучающей выборки; обучающая выборка зашумляется, тестовая выборка не изменяется.

Такая постановка была выбрана для того, чтобы продемонстрировать устойчивость проверяемых вариаций базовой системы как к искусственному, так и к естественному шуму.

4.3.2 Варианты системы распознавания именованных сущностей

Прежде всего, следует описать более подробно варианты инициализации матрицы векторных представлений слов.

Матрица векторных представлений обозначена далее, как матрицу $E \in \mathbb{R}^{V \times h}$, где V - это размер словаря (все уникальные слова, без добавления искусственного шума), а h – размерность векторного представления. Каждое слово представлено матричным умножением E и одинарным вектором $w \in \mathbb{R}^V$. В проведенных экспериментах $h = 128$.

RandomEmbed - случайно инициализируемая, не обучаемая матрица векторных представлений.

Системы fastText и Word2Vec, используемые для инициализации матрицы векторных представлений слов, брались публично доступные, предварительно обученные на больших корпусах. Для Word2Vec это была система, обученная на Google News, размерность векторного представления в этой системы - 300 измерений. Для fastText была взяла система fastText, обученная на Common Crawl. Размерность выходного вектора этой системы также составляет 300 измерений для векторного представления слова.

Список вариантов системы распознавания именованных сущностей

Таким образом, были протестированы 8 модельных систем с одномерным сверточным модулем в качестве обработки последовательности:

– **EmbedMatrix+CNN** – матрица векторных представлений слов и CNN уровня символов; матрица векторных представлений выучивается в процессе обучения в этой системе;

- **EmbedMatrix-nochar** – матрицы векторных представлений слов, без добавления признаков от побуквенного представления слова;
- **Word2Vec+CNN** – инициализация Word2Vec для матрицы векторных представлений слов и CNN уровня символов; матрица векторных представлений инициализируется векторными представлениями Word2Vec и не изменяется в процессе обучения в этой системе;
- **Word2Vec-nochar** – инициализация Word2Vec для матрицы векторных представлений слов, не используются признаки уровня символов;
- **FastText+CNN** – инициализация fastText для матрицы векторных представлений слов и функций, CNN на уровне символов; матрица встраивания инициализируется векторными представлениями из системы fastText и не изменяется в процессе обучения в этой системы;
- **FastText-nochar** – инициализация fastText для матрицы векторных представлений слов, не используются признаки уровня символов;
- **RandomEmbed+CNN** – инициализация матрицы векторных представлений слов слов случайно и CNN уровня символов; матрица векторных представлений инициализируется случайным образом и не изменяется в процессе обучения в этой системы;
- **RandomEmbed-nochar** – инициализация матрицы векторных представлений слов слов случайно, не используются признаки уровня символов.

Дополнительно системы EmbedMatrix+CNN, EmbedMatrix-nochar, FastText+CNN, FastText-nochar были протестированы со слоем, состоящим из ячеек LSTM, в качестве блока обработки последовательности. Эти системы были выбраны для этой дополнительной проверки, так как они показали лучшие результаты в основных экспериментах.

4.4 Результаты для задачи распознавания именованных сущностей

Результаты обучения и тестирования систем без проверки орфографии или искусственного шума представлены ниже. Каждая система оценивалась 10 раз при каждом уровне шума. Максимальное стандартное отклонение F_1 результат для данных CONLL был 0.060 (Word2Vec-nochar системы), для Persons-1000:

Система	CoNLL'03	Persons-1000	CAp'2017
EmbedMatrix+CNN	0.81	0.85	0.43
EmdebMatrix-nochar	0.80	0.81	0.44
RandomEmbed+CNN	0.69	0.77	0.31
RandomEmbed-nochar	0.61	0.48	0.22
FastText+CNN	0.86	0.69	0.41
FastText-nochar	0.86	0.69	0.41
Word2Vec+CNN	0.73	0.72	н/д
Word2Vec-nochar	0.72	0.72	н/д

Таблица 9 — результаты экспериментов с исходными наборами данных. Метрика F_1 на тестовой выборке.

0.105 (Word2Vec+CNN) и для CAp'2017: 0.066 (EmbedMatrix-nochar). Все результаты статистически значимы, p-значение < 0.05.

Результаты в таблице 9 представлены для сравнения со следующими результатами в шумной среде.

Результаты по всем наборам данных довольно схожи, что доказывает, с одной стороны, универсальность современной архитектуры biLSTM+CRF, а с другой стороны, влияние шума универсальности на разные языки. Можно упомянуть такое поведение на зашумленных тестовых наборах (Рис. 4.2, 4.3, 4.4) близко к линейной зависимости.

Отсутствие признаков символьного уровня не снижает устойчивость к шуму систем векторных представлений Word2Vec и fastText, но существенно влияет на системы с обучаемой матрицей векторных представлений, и эта зависимость сильнее проявляется для морфологически богатого русского языка. Кроме того, признаки уровня символов ожидаемо важны для случайных векторных представлений. Этот факт может быть истолкован в том ключе, что случайное векторное представление, служащее только для дифференциации конкретных слов, а символьные функции служат для сопоставления словоформ, в отличие от, например, fastText, где сами векторные представления слов устойчивы при наличии малого количества опечаток.

Графики тестирования качества систем для тестовых выборок, к которые не квносилось искусственного шума (рис. 4.5, 4.6, 4.7), показывают неожиданную устойчивость системы EmbedMatrix+CNN. Она показывает себя значительно лучше любой другой системы, практически не теряя качества

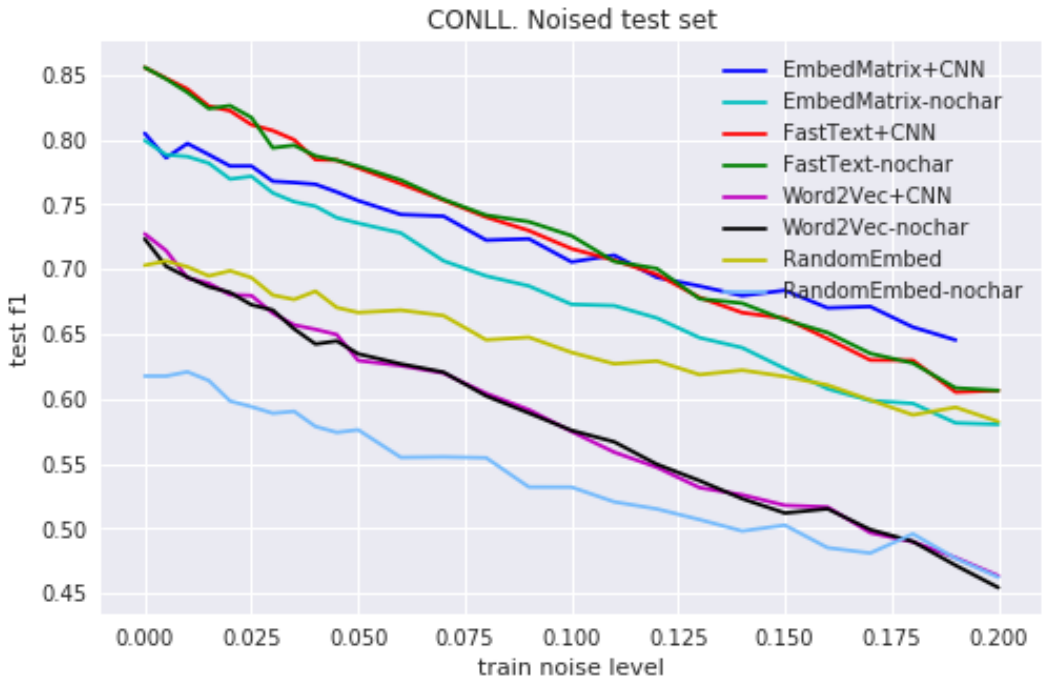

Рисунок 4.2 — Набор данных CoNLL'03. Обучение на очищенных и зашумленных данных, проверка на очищенных и зашумленных данных с тем же уровнем шума, что и обучающая выборка.

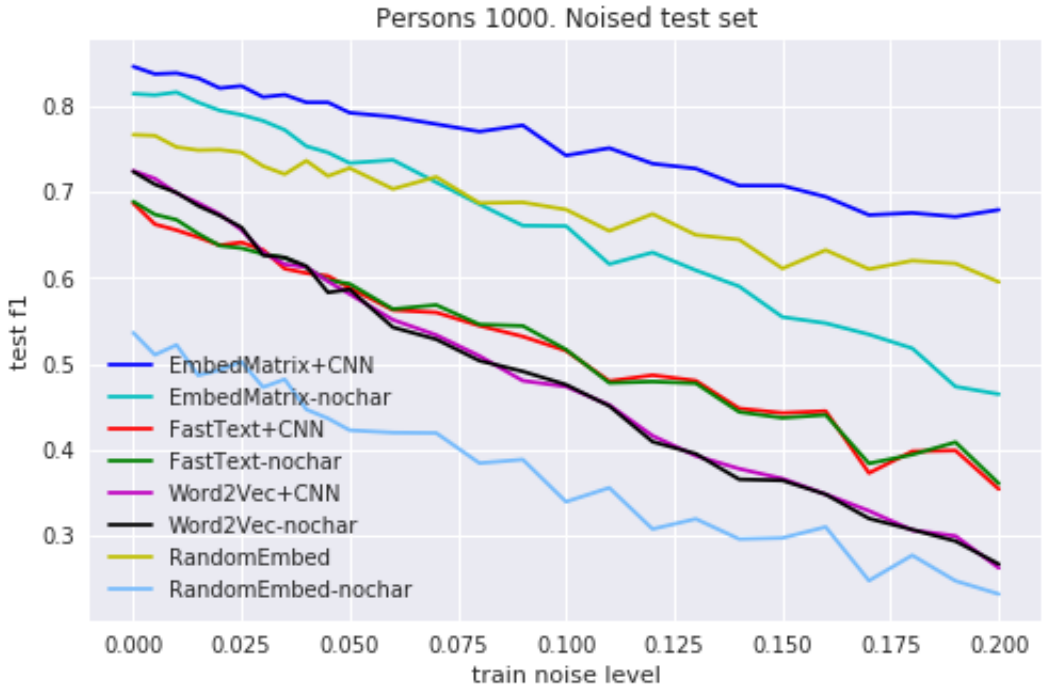

Рисунок 4.3 — Набор данных Persons-1000. Обучение на очищенных и зашумленных данных, проверка на очищенных и зашумленных данных с тем же уровнем шума, что и обучающая выборка.

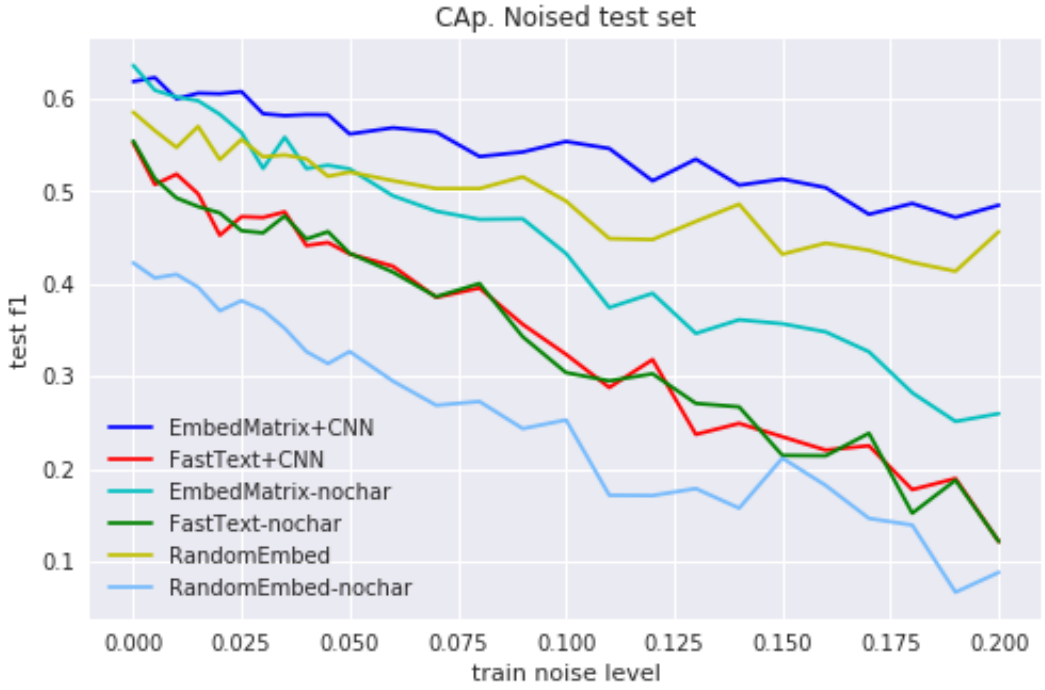

Рисунок 4.4 — Набор данных CAp'2017. Обучение на очищенных и зашумленных данных, проверка на очищенных и зашумленных данных с тем же уровнем шума, что и обучающая выборка.

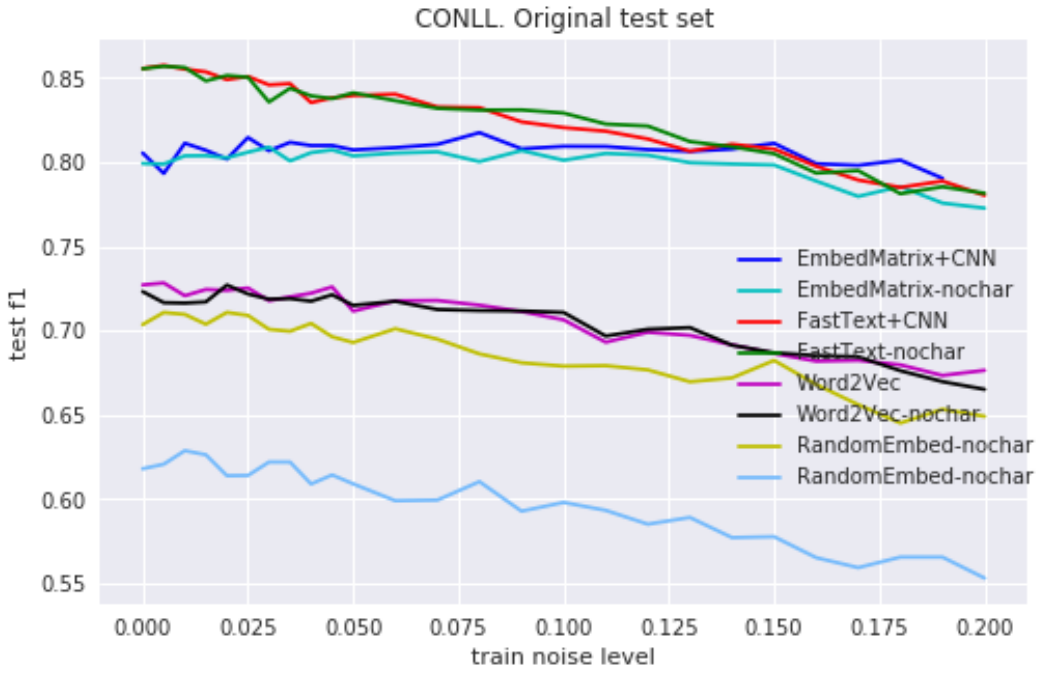

Рисунок 4.5 — Набор данных CoNLL'03. Обучение на очищенных и зашумленных данных, тестирование на неизмененных данных.

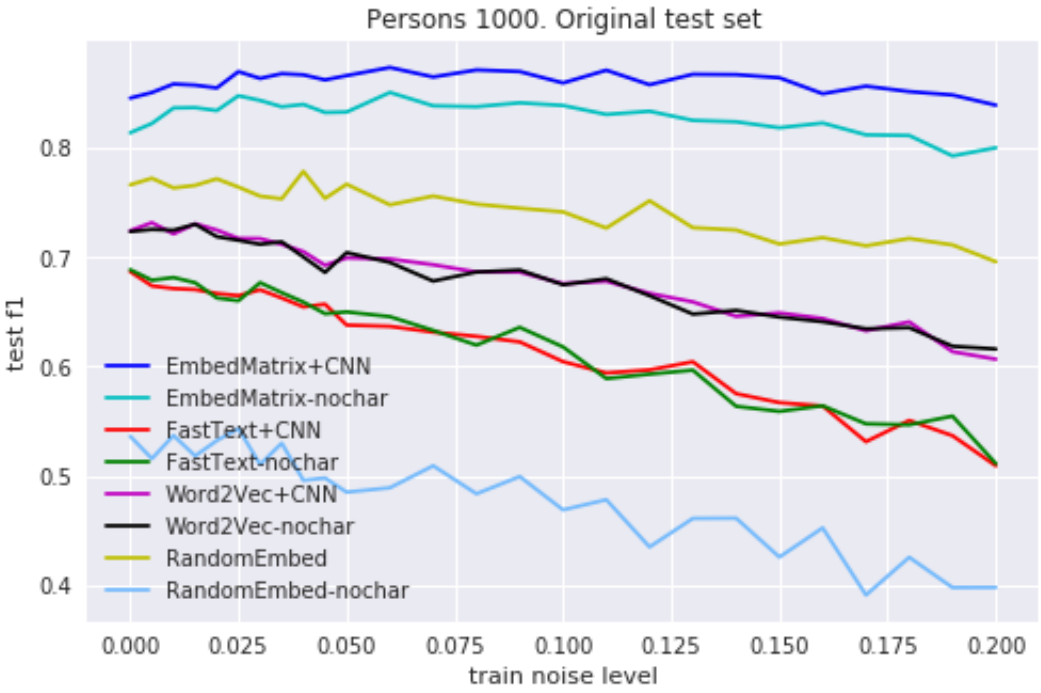

Рисунок 4.6 — Persons-1000 Dataset. Обучение на очищенных и зашумленных данных, проверка на неизмененных данных.

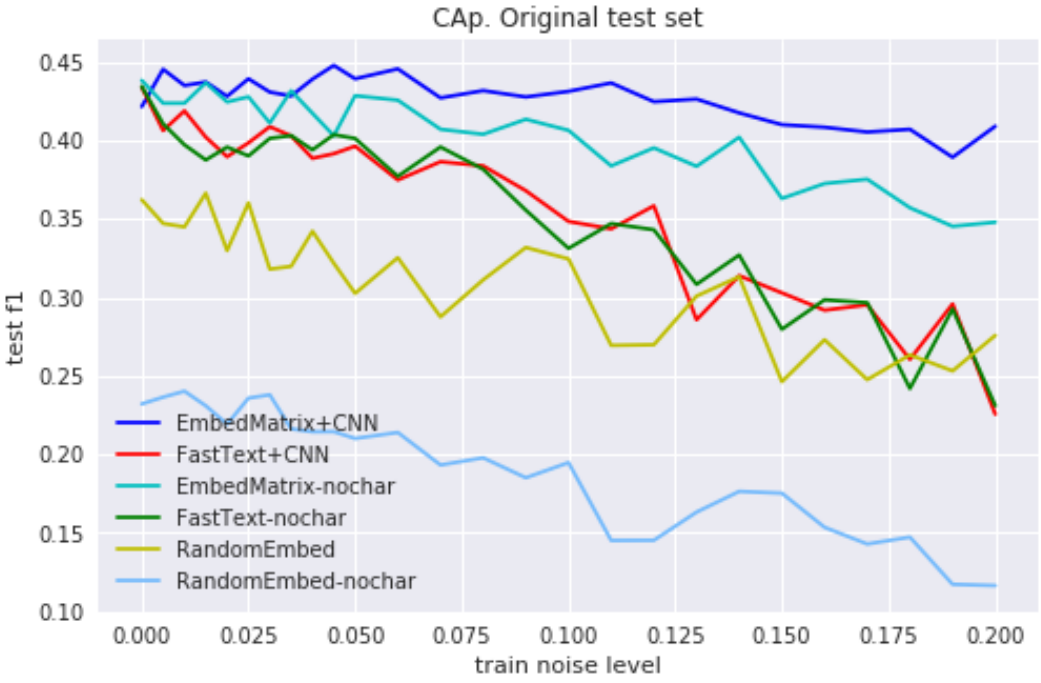

Рисунок 4.7 — CAp'17 набор данных. Обучение на очищенных и зашумленных данных, проверка на неизмененных данных.

при уровне шума от 0 до 0.2. Заметно, что признаки символьного уровня имеют существенное значение только для системы с обучаемой матрицей векторных представлений и более важны для русского и французского языков.

Системы fastText и Word2Vec остаются линейно зависимыми от уровня шума, при этом абсолютные значения немного меняются, а система Word2Vec превосходит систему fastText на наборе данных Persons-1000.

Такая эффективность обучаемой матрицы векторных представлений может быть объяснена тем, что это единственная система с векторными представлениями, обучаемыми совместно с системой распознавания именованных сущностей.

Кроме того, результаты для систем с модулем обработки контекста, основанном на ячейках LSTM, указывают на то, что устойчивость зависит гораздо больше от инициализации матрицы векторных представлений, чем от типа блока обработки последовательности (рис. 4.9, 4.8, 4.10).

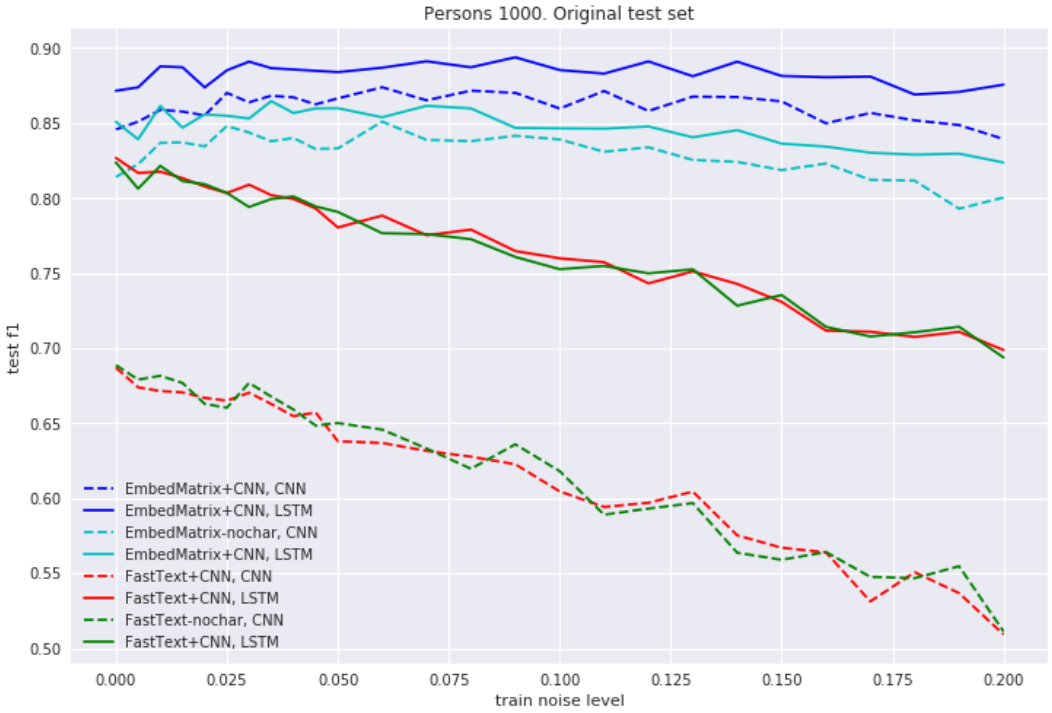

Рисунок 4.8 — Сравнение результатов систем с модулями обработки контекста, основанными на CNN и LSTM, на наборе данных Persons-1000.

Важно отметить, что на русском и французском наборах данных случайные векторные представления с признаками на уровне символов превосходят все векторные представления слов, кроме обучаемых векторных представлений (матрицы в терминах этой работы). Кроме того, на зашумленной тестовой

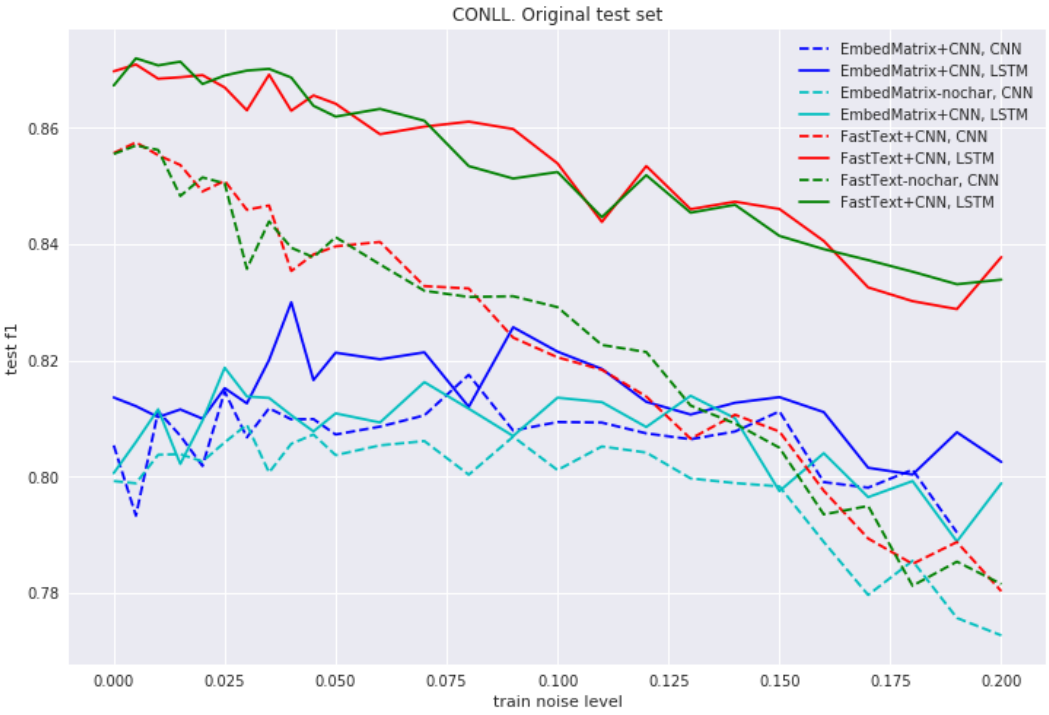

Рисунок 4.9 — Сравнение результатов систем с модулями обработки контекста, основанными на CNN и LSTM, на наборе данных CoNLL'03.

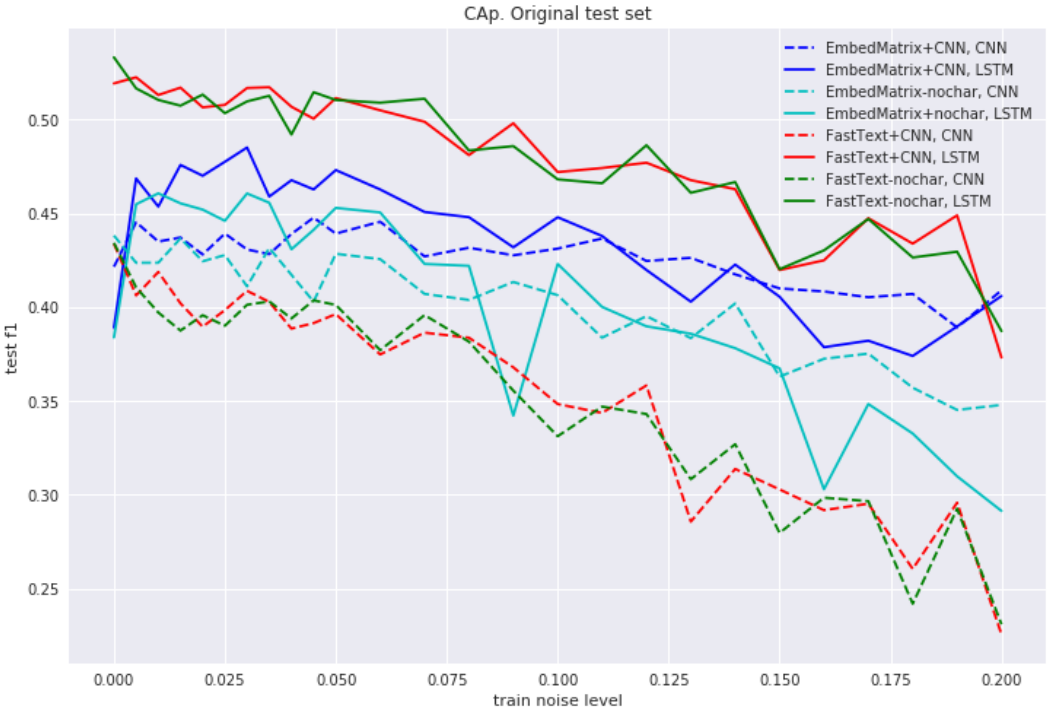

Рисунок 4.10 — Сравнение результатов систем с модулями обработки контекста, основанными на CNN и LSTM, на наборе данных CAp'2017.

Система	ориг. корпус	с пров. орфографии
EmbedMatrix+CNN, CNN	0.42	0.63
EmbedMatrix-nochar, CNN	0.44	0.64
EmbedMatrix+CNN, LSTM	0.39	0.59
EmbedMatrix-nochar, LSTM	0.38	0.59
FastText+CNN, LSTM	0.52	0.67
FastText-nochar, LSTM	0.53	**0.69**

Таблица 10 — Результаты экспериментов с набором данных CAp'2017. Метрика F_1 для очищенного и незашумленного корпуса.

выборке (рис. 4.3 и 4.4) система со случайными векторными представлениями даже превосходит систему с обучаемыми векторными представлениями на определенном (высоком) уровне шума.

Наконец, результаты показывают, что автоматическая проверка орфографии может иметь решающее значение в качестве предварительной обработки. Предыдущий лучший результат в соответствии с [106] составляет 58.59% F_1. Лучший результат предложенной системы по этому набору данных - 69.00% F_1 для системы FastText-nochar с модулем обработки контекста, основанным на ячейках LSTM.

4.5 Выводы к главе 4

В представленных экспериментах была продемонстрирована устойчивость лучшей современной системы biLSTM-CRF к шуму в виде опечаток в словах. Во многих экспериментах было показано превосходство выучиваемых векторных представлений, но эти векторные представления являются достаточно плохо обобщаемыми на реальные задачи. Векторные представления системы fastText обобщаются на реально встречаемый шум в данных, что и было продемонстрировано в экспериментах с неизмененной обучающей выборкой.

Неожиданно был достигнут новый результат на наборе данных на французском языке (без шума), поэтому важно отметить, что проверка орфографии важна для качества в задаче извлечении именованных сущностей на этом наборе данных данных.

Направления будущей работы автор видит в следующем: эксперименты над данными социальных сетей для английского и русского языков, включение немецких данных из набора данных CoNLL'03, тестирование других современных систем для задачи распознавания именованных сущностей, а также более глубокое исследование свойств шума, влияющих на качество системы.

Глава 5. Извлечение аспектов в шумных данных

В настоящее время задача извлечения аспектов становится все более популярной. Пользовательские термины часто пишутся с ошибками в шумных текстовых источниках, таких как комментарии в социальных сетях [2], например, из-за их сложной морфологии, что может привести к тому, что какие-то аспекты пользовательского сообщения не будут выделены. Это послужило мотивацией к изучению устойчивости модели ABAE с помощью искусственно созданного шума. В качестве базового решения была взята классическая модель латентного размещения Дирихле (Latent Dirichlet Allocation, LDA), описанная в работе Бляя [68].

В качестве модели шума была взята замена символов с некоторой вероятностью, т. е. для любой заданной строки эта строка рассматривается символ за символом и “бросается игральную кость”, нужно ли заменить эту конкретную букву ввода каким-то случайным символом.

Основные результаты работы:

– проведены эксперименты по устойчивости к шуму системы LDA;
– проведены эксперименты по устойчивости к шуму системы ABAE;
– предложены расширения системы ABAE, которые более устойчивы к шуму.

5.1 Система извлечения аспектов на основе внимания (ABAE)

Система извлечения аспектов на основе внимания (Attention-Based Aspect Extraction, ABAE) была предложена в работе Хе и соавторов [69]. Как и в тематическом моделировании или кластеризации, с помощью ABAE можно установить конечное число тем / кластеров / аспектов, и цель состоит в том, чтобы определить для каждого документа, в какой степени он соответствует любой из тем/аспектов.

Эта система представляет собой в своей основе автокодировщик, то есть систему, которая восстанавливает на выходе то, что было подано ей на вход с помощью промежуточного (сжимающего) представления. Главной его осо-

бенностью является функция потерь, основанная на близости порождаемого векторного представления входного предложения на основе векторных представлений слов и линейной комбинацией аспектных векторных представлений. Векторное представление предложения в данной системе - это не простое усреднение, как в модели мешка слов, здесь используется взвешивается так называемым само-вниманием, механизмом, близким к описанному в главе 1, для которого входными значениями являются векторные представления слов предложения, а ключом - средний вектор от векторов слов предложения.

Первый шаг в ABAE – построить векторное представление $z_s \in \mathbb{R}^d$ для предложения s:

$$z_s = \sum_{i=1}^{n} a_i e_{w_i} \tag{5.1}$$

где e_{w_i} - векторное представление $e \in \mathbb{R}^d$ для слова w_i. В качестве векторных представлений слов в исходной модели используются векторные представления word2vec модели CBOW [6]. Веса внимания a_i вычисляются как мультипликативная модель т.н. само-внимания (self-attention):

$$a_i = \text{softmax}(e_{w_i}^{\text{T}} \cdot A \cdot y_s) \tag{5.2}$$

$$y_s = \frac{1}{n} \sum_{i=1}^{n} e_{w_i} \tag{5.3}$$

где y_s среднее арифметическое от векторных представлений слов, входящих в предложение, а $A \in \mathbb{R}^{d \times d}$ - выучиваемая модель внимания.

Вторым шагом является вычисление аспектного векторного представления $r_s \in \mathbb{R}^d$ из аспектной матрицы матрицы $T \in \mathbb{R}^{k \times d}$, где k - количество аспектов:

$$p_s = \text{softmax}(W \cdot z_s + b) \tag{5.4}$$

$$r_s = T^{\text{T}} \cdot p_s \tag{5.5}$$

где $p_s \in \mathbb{R}^k$ - вероятности (веса) k векторов аспектных векторных представлений, и $W \in \mathbb{R}^{k \times d}$, $b \in \mathbb{R}^k$ – параметры многоклассовой системы классификации, логистической регрессии.

Для получения ошибки восстановления в системе используется косинусное расстояние между r_s и z_s с контрастивной целевой функцией max-margin, описанной в работе Вестона и соавторов [107]. Кроме того, к целевой функции

добавляется штраф за ортогональность, который принуждает матрицу векторного представления аспектов T производить как можно более разнообразные векторные представления аспектов. Вся архитектура системы целиком представлена на рис. 5.1.

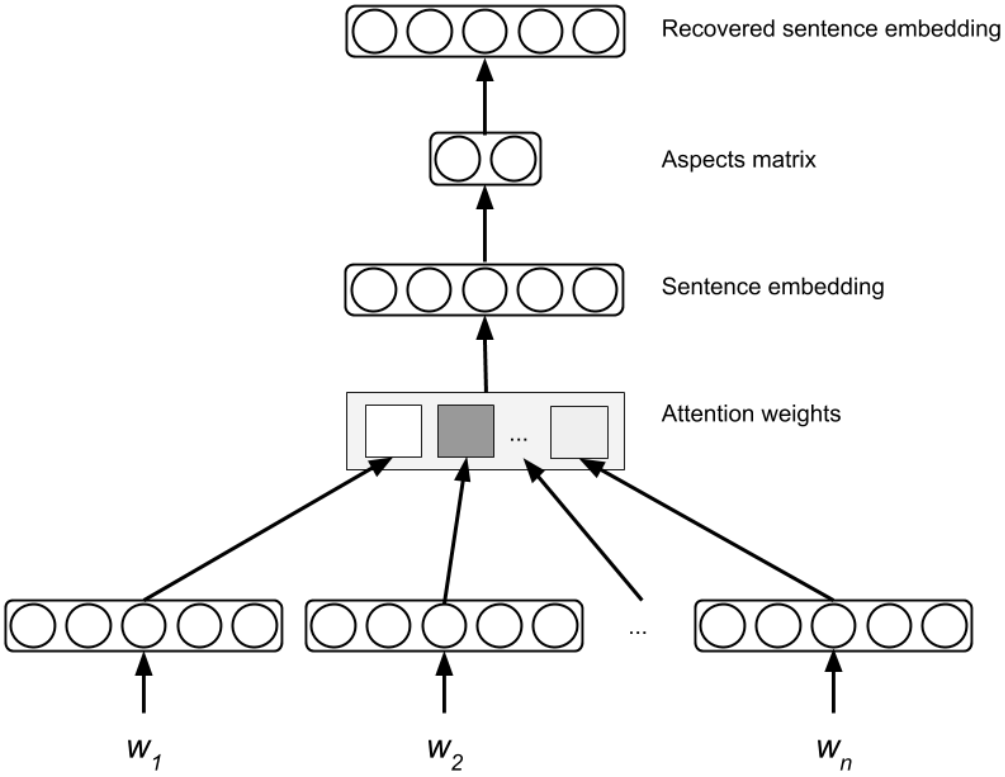

Рисунок 5.1 — Архитектура системы ABAE

5.2 Модификации системы ABAE

Метод построения систем, устойчивых к шуму, в задаче извлечения аспектов

Система извлечения аспектов может быть представлена как совокупность модулей векторного представления слов и сопоставления аспектов. *Метод создания программных систем* состоит в модификации модуль векторного представления слов таким образом, чтобы он был устойчив к опечаткам. Модуль векторного представления слов может состоять из двух компонент -

компоненты векторного представления слов на основе символов и компоненты представления слов, как целого. Для того, чтобы придать системе свойство устойчивости к шуму в рамках предлагаемого *метода построения программных систем* необходимо, аналогично главе 2: а) выделить в системе компоненты буквенного представления слов и представления контекста, б) описать взаимодействие этих модулей и входных данных, в) описать процедуру обучения модуля контекстного представления на обучающей выборке. Модуль буквенного представления строится таким образом, чтобы быть мало- или нечувствительным к шуму разного рода. Модуль обработки контекста обеспечивает сохранение семантических свойств при использовании модуля буквенного представления.

Интуиция предлагаемых изменений заключается в том, что орфографические ошибки в данных можно некоторым способом проигнорировать. Один из способов сделать это - сделать векторные представления слов (компоненту представления слов) устойчивыми к шуму в написании слов. Были выбраны три основных способа обработки такого шума.

В работе Миколова и соавторов [28] авторами предложена система векторного представления слов, где векторное представление слов генерируется как сумма вложений, состоящих из n-граммов. Процедура обучения следующая: система берет векторные представления для всех n-грамм слова и суммирует их, используя этот результирующий вектор система предсказывает все окружающие слова в контексте с помощью SoftMax. Эта система содержит ограничение, что если слово не имеет известных для модели n-грамм, то система не может создать векторное представление для него. Но, тем не менее, эта система широко используется на практике. Модификация системы ABAE, в которой в качестве векторных представлений слов берутся представления, порождённые системой fastText, ниже по тексту называется **FastText**.

Другой подход был предложен в работе Кима и соавторов [60] для классификации предложений, но может быть легко адаптирован для задачи извлечения аспектов. Ключевая идея заключается в использовании сверточных нейронных сетей над векторными представлениями символов для векторного представления слов. Для того, чтобы произвести более эффективное векторное представление, в экспериментах используются как векторные представления слов, так и векторные представления символов, которые объединяются в конечный вектор для слова. Эта система была предложена в работе Ли и Хуанга [108].

Модификация системы ABAE, в которой в качестве векторных представлений слов берутся представления, получаемые на основе векторных представлений символов, ниже по тексту называется **CharEmb**.

Последний рассматриваемый подход был предложен в работе автора [11]. В данной работе автором представлена система, порождающую векторные представления слов на основе буквенного представления слова (этот подход близок к подходу “мешок букв”, bag of characters) и контекста. Модификация системы ABAE, в которой в качестве векторных представлений слов берутся представления, порождённые системой RoVe, ниже по тексту называется **RoVe**.

5.3 Эксперименты по сравнению систем извлечения аспектов

В качестве меры качества была выбрана метрика F_1, взвешенная по классам, как и в работе, описывающей исходную систему ABAE [69]. Для того, чтобы осуществить оценку качества по этой метрике необходимо иметь правильные метки и метки, предсказанные системой. В качестве правильных меток были взяты размеченные категории аспектов для так называемого “ресторанного” домена (корпус Citysearch с отзывами пользователей о местных предприятиях, первоначально описанный в работе Гану [109]). В этом корпусе каждому отзыву сопоставляется одна или несколько категорий аспектов, таких как “еда” (Food) или “атмосфера” (Ambience). Из корпуса были отобраны только те отзывы, которые содержат ровно одну категорию. Корпус был разбит на обучающую и тестовую выборки, разбиение было опубликовано Хе и соавторами в работе [69]. Модель была натренирована на обучающей выборке из описанного корпуса и выделенные аспекты были вручную размечены на соответствие категориям. Для получения предсказанных меток на тестовой выборке в качестве предсказания модели брался максимально вероятный аспект, которому сопоставлялась ранее размеченная категория. Примеры, выделяемых системой ABAE аспектов можно увидеть в таблице 11.

Для сравнения была взята система на основе латентного размещения Дирихле, описанного в работе Бляя [68].

Категория	Характерные слова
Food	beef, duck, pork, mahi, filet, veal gelato, banana, caramel, cheesecake, pudding, vanilla bottle, selection, cocktail, beverage, pinot, sangria cucumber, scallion, smothered, stewed, chilli, cheddar cooking, homestyle, traditional, cuisine, authentic, freshness
Ambience	wall, lighting, ceiling, wood, lounge, floor intimate, comfy, spacious, modern, relaxing, chic
Staff	waitstaff, server, staff, waitress, bartender, waiter unprofessional, response, condescending, aggressive, behavior, rudeness
Price	charge, paid, bill, reservation, came, dollar
Anecdotes	celebrate, anniversary, wife, fiance, recently, wedding
Misc.	park, street, village, avenue, manhattan, brooklyn excellent, great, enjoyed, best, wonderful, fantastic aged, reward, white, maison, mediocrity, principle

Таблица 11 — Примеры аспектов, получаемых моделью ABAE на корпусе Citysearch.

5.3.1 Метод сравнения систем извлечения аспектов

Метод сравнения программных систем на предмет устойчивости к шуму состоит из следующих этапов: а) этапа контролируемого внесения шума во входные данные, б) этапа проверки качества тестируемой системы. Для того, чтобы проверить качество некоторой системы при заданном уровне шума необходимо внести во входные данные для тестирования системы шум заданного уровня и проверить качество вывода системы относительно известной разметки для тестовых данных. Данную процедуру необходимо провести для каждой системы и каждого уровня шума. Каждый такого рода эксперимент необходимо повторить некоторое количество раз для статистической достоверности получаемых результатов. В описываемой главе каждый эксперимент был повторен 10 раз, полученные результаты имеют достоверность $p < 0.05$.

В качестве метода сравнения систем предлагается метод внесения шума в тестовые данные. Для сравнения системам, обученным на чистых, незашумленных данных подается на вход текст с внесенным шумом и замеряется качество

на некоторой задаче. Чтобы продемонстрировать устойчивость к шуму, в рассматриваемый корпус был искусственно добавлен шум. Этот шум описывается следующим образом:

– вероятность замены буквы случайной буквой из алфавита

для каждой буквы входного текста. Для вставки символа, требуемый символ порождается из алфавита текста равновероятно. На каждой входной строке выполняются случайные вставки и удаления букв с моделируемыми вероятностями. Системы были протестированы с различным уровнем шума от 0% (без шума) до 30%. Согласно работе Кучержана и Брилла [2], реальный уровень шума в генерируемых пользователем текстах составляет 10-15%.

5.3.2 Постановка экспериментов для задачи извлечения аспектов

Для инициализации аспектной матрицы $\mathbf{T}$ в системе ABAE была взята система Word2Vec [6], натренированная на корпусе Citysearch. Количество аспектов было установлено в 14, следуя работе Xe и соавторов [69].

Для системы векторных представлений fastText используется публично доступная система fastText для английского языка. [1]

Для системы CharEmb используется предварительно обученная на наборе документов сервиса Google News система Word2Vec.[2]

Система RoVe была обучена на наборе данных Reuters, описанном в работе Льюиса и соавторов [91], с двунаправленным кодировщиком SRU, описанном в работе Ли и соавторов в работе [79], и фиксированной длиной равной 3 для начальной и конечной частей представления слова.

Для системы LDA количество аспектов также было установлено в 14, как и для модели ABAE. Параметр $\tau_0 = 50$. По сравнению с классическим алгоритмом, был выбран иной метод обучения, а именно вариационный Байесовский, так как он позволяет производить обучение на больших объемах данных быстрее. Эти параметры были выбраны так как показали лучшее качество в предварительных экспериментах.

[1] https://fasttext.cc/docs/en/english-vectors.html

[2] Эта система доступна для скачивания отсюда.

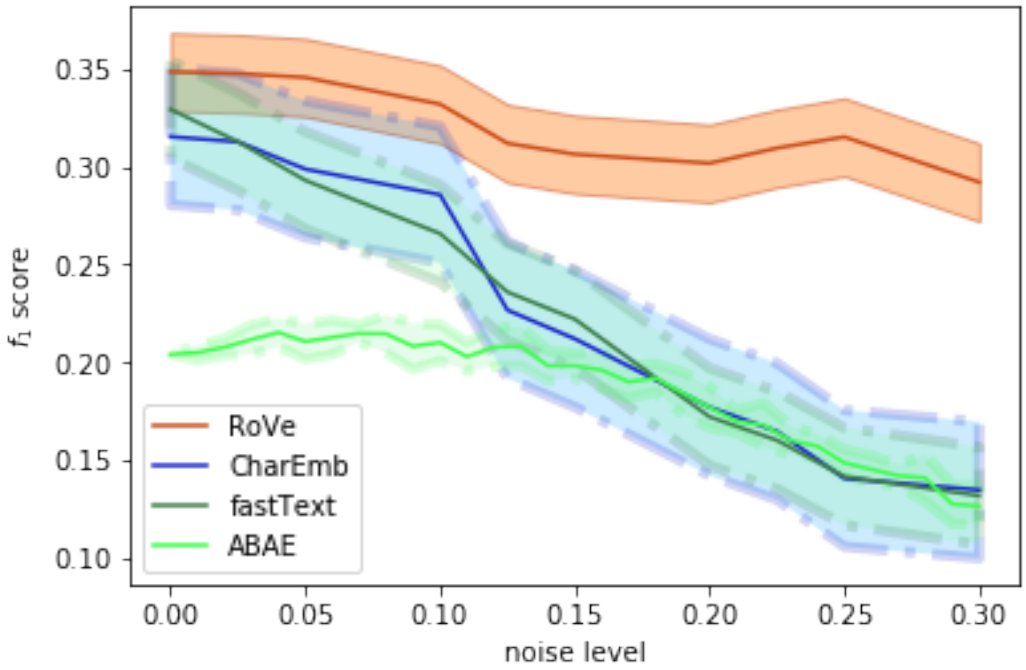

Рисунок 5.2 — Качество по метрике F_1 для исходной программной системы ABAE и предлагаемых модификаций.

5.4 Результаты экспериментов для задачи извлечения аспектов

Во-первых, были проведены эксперименты с исходной системой ABAE. Поскольку авторы предложили использовать Word2Vec, ожидается, что этот подход не будет иметь устойчивости к опечаткам. В системе Word2Vec отсутствует поддержка несловарных слов (OOV) и для шумного входа эта система не будет способна выдавать векторные представления для всех слов. На практике в случае, если система не способна выдать векторное представлние для слова, считается, что она порождает нулевой вектор. Рис. 5.2 показывает, что качество системы неожиданным образом улучшается при низких значениях уровня шума по сравнению с исходными (незашумленными) данными. Это может быть объяснено тем, что используемая программная система Word2Vec была обучена на текстах, содержащих некоторый естественный шум.

Во-вторых, были проведены эксперименты с описанными модификациями системы ABAE. Как видно из рис. 5.2 хотя все предложенные системы более устойчивы к шуму, чем оригинальная система, ни одна из них не является нечувствительным к шуму; системы FastText и CharEmb очень близки по своему качеству на всех уровнях шума, в отличие от системы RoVe, которая показы-

вает не только лучшее качество среди предложенных модификаций системы ABAE на всех уровнях шума, но и большую устойчивость в целом, так как теряет качество медленнее прочих систем.

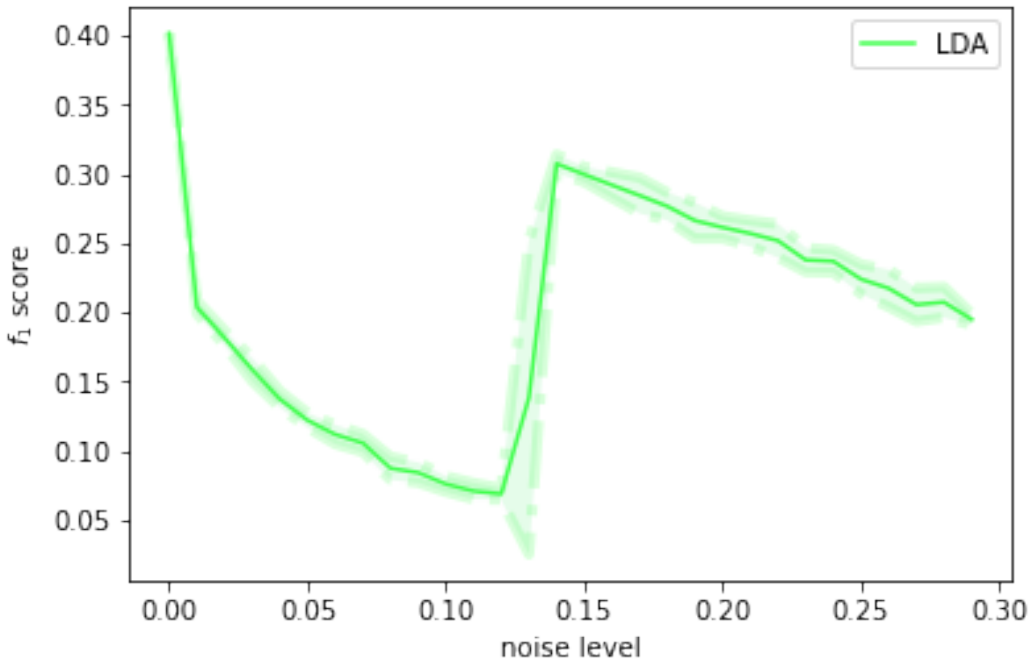

Рисунок 5.3 — Качество по метрике F_1 для системы LDA.

В-третьих, было произведено тестирование базовой системы LDA. Полученные результаты приведены на рисунке 5.3. Данная система показала хорошее качество в рассматриваемой задаче для нулевого уровня шума, но очень быстро теряет в качестве с добавлением даже небольшого шума в данные. Интересно поведение системы на уровне шума 0,15 - модель имеет резкий скачок в качестве. В дальнейшем уровень шума снижается аналогично участку $[0; 0.10]$. При сравнении с программной системой RoVe, качество на нулевом уровне шума у системы LDA меньше, но на прочих уровнях шума эта система проигрывает системе RoVe.

5.5 Выводы к главе 5

Разработанные модификации программной системы ABAE показывают улучшение относительно исходной системы, в дополнение к этому система RoVe имеет лучшую устойчивость в шуму. Последний факт можно интерпретировать

как то, что изначально закладываемая в архитектуру системы устойчивость к шуму является эффективным средством по сохранению качества работы системы в условиях шумного входа. В некоторых случаях можно наблюдать улучшение устойчивости к шуму. Программная система LDA показывает лучшее качество на нулевом шуме, но на малых уровнях шума быстро теряет качество, что объяснимо тем, что для этой модели требуется полное совпадение написания слова.

В качестве будущей работы автор рассматривает следующие направления:

- существуют другие типы шума, которые могут быть протестированы в рамках используемого метода тестирования;
- существуют другие языки, например, с богатой морфологией, где устойчивость к шуму может быть более важной особенностью, чем для английского языка;
- существуют другие системы извлечения аспектов, основанные на других подходах, которые могут быть устойчивыми к шуму режима.

Глава 6. Комплекс программ для оценки устойчивости к шуму систем для задач обработки текстов

Данная глава посвящена оценке алгоритмической сложности разработанных программных комплексов. В подглаве 6.1 приводятся оценки сложности программных систем, описанных в главе 2. В подглаве 6.2 приведены оценки сложности для систем классификации текстов, описанных в главе 3. В подглаве 6.3 приведены оценки сложности для распознавания именованных сущностей, описанных в главе 4. В подглаве 6.4 приведены оценки сложности для систем классификации текстов, описанных в главе 5.

В рамках данной диссертационной работы был разработан программный комплекс по построению и сравнению систем векторных представлений слов, классификации текстов, распознаванию именованных сущностей и извлечению аспектов. Данный комплекс частично выложен в открытый доступ[1] и включает в себя следующие условно-независимые пакеты программ:

- пакет программ для построения и сравнения систем векторных представлений слов;
- пакет программ для построения и сравнения систем классификации текстов;
- пакет программ для построения и сравнения систем распознавания именованных сущностей;
- пакет программ для построения и сравнения систем извлечения аспектов.

Данные пакеты связаны между собой следующим образом: пакет программ для построения векторных представлений слов порождает систем векторных представлений слов, которые используются остальными пакетами программ.

Все программные пакеты используют следующие процедуры генерации шума в данных: 1, 2, 3, 4. Стоит отметить, что процедуры генерации шума 3 и 4 могут быть заменены на комбинацию из 1 и 2, но в силу того, что такого рода опечатки встречаются на практике, то они моделируются как отдельные виды шума.

Сложность процедуры 1 составляет $O(2n)$ в худшем случае, где n - длина входной строки. Сложность процедуры 2 составляет $O(3n)$ в худшем случае.

[1] https://gitlab.com/madrugado/robust-w2v

Сложность процедуры 3 составляет $O(3n)$ в худшем случае. Сложность процедуры 4 составляет $O(3n)$ в худшем случае.

Для дальнейших оценок необходимо ввести несколько обозначений:

L_w - средняя длина слова в тексте.

L - длина текста в символах.

l - длина текста в словах.

k - выходная размерность системы.

d - входная размерность системы.

Величины L, L_w и l связаны следующим соотношением:

$$l = L/L_w.$$

Input: Строка S, уровень шума p
Output: Строка Z
$Z \leftarrow \varnothing$
for *Символ C из строки S* **do**
  генерация числа $d \in [0,1]$
  if $d > p$ **then**
    скопировать символ C в конец строки Z
  end
end

Процедура 1: Процедура генерации шума путем удаления части символов.

Данные для подачи на обучение для всех пакетов программ пропускаются через процедуру внесения шума, которая обрабатывает входной текст посимвольно. После получения зашумленных данных, полученные данные разбиваются на токены, используя символ пробел (“ ”). Сложность этой операции составляет $O(n)$, т.к. она представляет собой однократный обход массива и сравнение каждого элемента с эталоном (символом пробела).

Для анализа сложности рассматриваемых систем будет использоваться то допущение, что функции сигмоиды и гиперболического тангенса являются элементарными. В общем случае это неверно, но сложность выполнения этих функций на современных процессорах, например, описанных в работе [110], является постоянной, не зависящей от входных параметров. Так что получаемые далее оценки верны с точностью до константного множителя.

Input: Строка S, уровень шума p, алфавит A
Output: Строка Z
$Z \leftarrow \varnothing$
for *Символ C из строки S* **do**
    генерация числа $d \in [0,1]$
    if $d > p$ **then**
        скопировать символ C в конец строки Z
    else
        скопировать символ C в конец строки Z
        выбрать символ K из A
        скопировать символ K в конец строки Z
    end
end

Процедура 2: Процедура генерации шума путем добавления символов.

Input: Строка S, уровень шума p, алфавит A
Output: Строка Z
$Z \leftarrow \varnothing$
for *Символ C из строки S* **do**
    генерация числа $d \in [0,1]$
    if $d > p$ **then**
        скопировать символ C в конец строки Z
    else
        выбрать символ K из A
        скопировать символ K в конец строки Z
    end
end

Процедура 3: Процедура генерации шума путем замены части символов.

Input: Строка S, уровень шума p
Output: Строка Z
Пусть L - длина строки S. $Z \leftarrow \varnothing$
for $i \in [1, L]$ **do**
  генерация числа $d \in$ [0,1]
  Пусть C - i-ый символ строки S, K - $(i-1)$-ый символ строки S
  if $d > p$ **then**
    скопировать символ C в конец строки Z
  end
  заменить последний символ строки Z символом C
  скопировать символ K в конец строки Z
end

Процедура 4: Процедура генерации шума путем перемешивания части символов.

Важно отметить, что в настоящей главе оценка сложности всех систем производится для случая запуска на тестирование, т.к. с соответствующих главах сравнение производится именно в этом режиме.

6.1 Пакет программ для построения и оценки устойчивости систем векторных представлений слов

6.1.1 Система RoVe

Система RoVe состоит из двух главных частей, модуля представления BME и модуля обработки контекста, которые рассмотрены ниже. Общая сложность системы RoVe для лучшего по результатам экспериментов модуля обработки SRU составляет

$$O(\frac{L}{L_w} \times (n_b + n_e + 1 + 3k \times d + 10k)).$$

Модуль BME

Алгоритм работы модуля BME представлен в разделе 2.3.1. Сложность работы этого алгоритма можно оценить как сумму сложностей работы отдельных его частей. Части алгоритма, генерирующие части B и E, имеют сложность $O(n_b)$ и $O(n_e)$ соответственно, то есть константную. Часть M содержит суммирование по длине слова, то есть имеет сложность $O(L_w)$, где L_w длина слова w. Если рассмотреть случай русского языка, для которого средняя длина слова составляет 5.2, то общая сложность работы алгоритма на тексте длины (в символах) L составит

$$O(\frac{L}{5.2} \times (n_b + n_e + 1)).$$

Модуль обработки контекста

Модуль обработки контекста представлен несколькими вариантами. Варианты могут быть условно поделены на две большие группы, в рамках которых сложность отличается несущественно - это варианты, построенные на рекуррентных сетях, и варианты, построенные на сверточных сетях.

Вариант SRU. Пусть k - выходная размерность ячейки SRU, а d - входная размерность. Тогда сложность обработки одного слова согласно уравнениям 2.2 составит

$$O(k \times d + k \times d + k + k + k \times d + k + k + k + k + k + k + k + k) = O(3k \times d + 10k),$$

то есть сложность получается линейная относительно фиксированных размерностей k и d. Если входная и выходная размерности равны, то общая сложность получает квадратичной относительно этой размерности.

Вариант biSRU. Пользуясь результатами предыдущего вычисления и учитывая тот факт, что в двунаправленной рекуррентной сети все вычисления удваиваются получаемая сложность составит

$$O(6k \times d + 20k).$$

Вариант cnn-1d. В силу особенности работы сверточных сетей, данный вид нейронных сетей может работать только с последовательностями фиксированной длины. Сложность работы этого варианта согласно уравнения 1.19 составляет для первого слоя

$$O((l - 3 + 1) \times 3 \times d),$$

для второго слоя

$$O((l - 2 - 5 + 1) \times 5 \times d),$$

и соответственно для третьего слоя

$$O((l - 6 - 3 + 1) \times 3 \times d).$$

Последний слой - это слой субдискретизации, его сложность составляет

$$O((l - 8) \times d).$$

Что дает суммарную сложность

$$O((11l - 60) \times d),$$

то есть линейную сложность относительно входной размерности d и длины последовательности l.

6.1.2 Система Word2Vec

Сложность системы Word2Vec, т.к. она является системой, основанной на словаре, равно сложности поиска в множестве упорядоченных элементов и зависит от реализации. Сложность поиска по упорядоченному множеству для оптимального случая составляет $O(1)$, например, для хэш-таблицы.

6.1.3 Система fasttext

Система fasttext является прямым расширением системы Word2Vec. Данная система помимо словаря слов содержит словарь n-грамм. Соответственно,

сложность получения вектора слова для данной системы составляет сложность поиска по словарю слов и дополнительно сложность поиска по словарю n-грамм. В зависимости от настроек системы fasttext, словарь n-грамм может содержать таковые различной длинны, типично от 1 до 4. Если предположить, что мы имеем дело в триграммной системой, наиболее часто встречающейся на практике, то в общем виде сложность системы fasttext будет $O(L_w - 3 + 1 + 1)$. Если предложить, что рассматривается система для английского языка, то, учитывая, что средняя длина слова в английском языке составляет 5.1, то сложность поиска в словаре n-грамм для типичного слова составит $O(3)$, что в сумме дает $O(4)$ или константную сложность.

6.2 Пакет программ для построения и оценки устойчивости систем классификации текстов

6.2.1 Система CharCNN

Данная система содержит сверточный слой и слой субдискретизации. Воспользуемся уравнением 1.19, чтобы получить сложность работы сверточного слоя:

$$O(\frac{(L - 15 + 1)}{2} \times d).$$

Для слоя субдискретизации сложность будет составлять

$$O((L - 14) \times d).$$

Итого общая сложность данной системы для текста длиной L символов составляет

$$O((1.5L - 21) \times d).$$

6.2.2 Система Fasttext-GRU

Данная система содержит в себе два модуля - векторных представлений слов fasttext 6.1.3, для которой сложность составляет $O(4)$, и модуля обработки текста. Модуль обработки текста представлен рекуррентной сетью типа GRU. Для данного типа рекуррентных сетей, согласно уравнениям 1.17, сложность будет составлять

$$O(k\times d+k\times k+k+k\times d+k\times k+k+k\times d+k+k\times k+k+k+k+k) = O(k\times 3d+3k^2+7k).$$

Соответственно для текста длиной l слов составит

$$O(l \times (k \times 3d + 3k^2 + 7k + L_w - 2)).$$

6.2.3 Система CharCNN-WordRNN

Система CharCNN-WordRNN содержит в качестве модуля обработки текста GRU, рассмотренный в разделе 6.2.2, а в качестве модуля векторного представления слов сверточную нейронную сеть, сложность которой может быть оценена с привлечением уравнения 1.19:

$$O((L_w - 5 + 1) \times d + (L_w - 4) \times d) = O((L_w - 4) \times 2d).$$

Следовательно, общая сложность составит

$$O((L_w - 4) \times 2d + k \times 3d + 3k^2 + 7k),$$

то есть квадратичной от размерности k.

6.2.4 Система RoVe

Система классификации RoVe сочетает в себе в качестве модуля векторного представления слов система векторного представления слов RoVe и в

качестве модуля обработки текста GRU. Используя результаты разделов 6.1.1 и 6.2.2, получим сложность данной системы

$$O(l \times (n_b + n_e + 1 + 3k_{RoVe} \times d_{RoVe} + 10k_{RoVe} + 3k_{GRU} \times d_{GRU} + 3k_{GRU}^2 + 7k_{GRU})).$$

Если положить, что все входные и выходные размерности равны k, то сложность упрощается до

$$O(l \times (n_b + n_e + 1 + 17k + 9k^2)).$$

6.3 Пакет программ для построения и оценки устойчивости систем распознавания именованных сущностей

6.3.1 Система LSTM-CRF

Сложность архитектуры LSTM-CRF, которая рассматривается в главе 4, можно оценить помодульно. Сначала будет рассмотрен модуль посимвольного представления. Данный система строится аналогично модулю посимвольного представления в разделе 6.2.3, таким образом его сложность составляет

$$O((L_w - 4) \times 2d).$$

Система векторного представления слов представлена 4 вариантами, а именно Word2Vec, fasttext, RandomEmbed и EmbedMatrix. Сложность модулей Word2Vec и fasttext была оценена в разделах 6.1.2 и 6.1.3 соответственно. Сложности модулей RandomEmbed и EmbedMatrix полностью аналогична Word2Vec, т.к. принцип получения векторного представления во всех трех рассматриваемых модулях аналогичен. Итоговая сложность векторного представления слов: Word2Vec - $O(1)$, fasttext - $O(4)$, RandomEmbed - $O(1)$, EmbedMatrix - $O(1)$.

Модуль обработки текста в рамках главы 4 рассмотрен двух видов: основанный на сверточных нейронных сетях и на рекуррентных, конкретно на долговременной краткосрочной памяти.

Для сверточной нейронной сети нужно опять воспользоваться уравнением 1.19. Получающая оценка аналогична модулю посимвольного представления:

$$O((l - 4) \times 2d).$$

Для рекуррентной сети нужно воспользоваться уравнениями 1.16. Получаемая сложность:

$$\begin{aligned} &O(k \times d + k \times k + k + k + k \times d + k \times k + k + k \\ &+k \times d + k \times k + k + k + k \times d + k \times k + k + k \\ &+k \times d + k \times k + k + k + k + k + k + k) \\ &=O(4k \times d + 4k^2 + 12k). \end{aligned}$$

Нужно отметить, что в описываемой архитектура используется двунаправленная рекуррентная сеть, таким образом, итоговая сложность модуля обработки текста составит

$$O(8k \times d + 8k^2 + 24k).$$

Модуль CRF имеет сложность, зависящую от используемых потенциальных функций 1.25. Типично на практике используются унарные и бинарные потенциальные функции, для этого случая и будет сделана оценка. Также сложность модуля CRF зависит от количества используемых меток t. Итоговая сложность модуля CRF в описываемом случае:

$$O((l-1) \times t^2 + l \times t).$$

Таким образом, для наиболее устойчивой в большинстве рассмотренных случаев системы, использующей посимвольное представление слов и использующей в качестве модуля векторного представления слов fasttext, а также модуль обработки текста LSTM общая сложность составит

$$O(\frac{l}{L_w}(L_w - 4) \times 2d + \frac{l}{L_w}(L_w - 2) + (l-4) \times 2d) + (l-1) \times t^2 + l \times t).$$

6.4 Пакет программ для построения и оценки устойчивости систем извлечения аспектов

В данном пакете программ реализованы оригинальная система ABAE и расширения к ней, а именно CharCNN, fasttext, RoVe.

Сложность системы ABAE состоит из модуля векторных представлений слов и остальной системы. Сложность системы ABAE зависит от количества

аспектов a. Сложность системы составляет согласно уравнениям 5.1, 5.3, 5.5:

$$O(l\times d+d+l\times(d^2+d)+l+l+k\times d+k+a\times k) = O(l\times(d^2+2d)+d+2l+k+a\times k+1).$$

Дополнительная константная сложность в этом уравнении отражает то, что в этой системы используется векторное представление слов Word2Vec.

Сложность модуля векторного представления слов **CharCNN** составит, пользуясь результатами раздела 6.2.3,

$$O((L_w - 4) \times d + 1).$$

Дополнительная константная сложность в этом уравнении отражает то, что в этом варианте модуля используется векторное представление слов Word2Vec.

Сложность модуля векторного представления слов **fasttext**, пользуясь результатами раздела 6.1.3, составляет $O(L_w - 2)$.

Сложность модуля векторного представления слов **RoVe**, пользуясь результатами раздела 6.1.1, $O(\frac{L}{L_w} \times (n_b + n_e + 1 + 3k \times d + 10k))$.

Таким образом, сложность системы ABAE с расширением RoVe составляет:

$$O(l\times(d^2_{ABAE}+2d_{ABAE})+d_{ABAE}+2l+k+a\times k+l\times(n_b+n_e+1+3d_{ABAE}\times d_{RoVe}+10_{ABAE})).$$

Здесь учтено, что выходная размерность RoVe является входной для системы ABAE.

6.5 Выводы к главе 6

На всех рассмотренных задачах сложность зависит квадратично от размерностей входных векторов. Также стоит отметить, что те варианты систем, которые демонстрируют лучшее качество, имеют тенденцию быть более сложными в теоретико-информационном смысле, например система ABAE против системы ABAE с расширением RoVe.

Заключение

Существующие программные системы создания векторных представлений слов в своей массе не рассчитаны на работу с шумными текстами. Вместо этого используются системы проверки орфографии. Система Word2Vec, описанные в работах Т. Миколова [6] и [25], полностью игнорирует существование опечаток в текстах, полагаясь на увеличение словаря системы, что очевидным образом имеет два недостатка - словарь сисемы разрастается и для обучения такой системы требуются все большие объемы данных. Для другой популярной системы fastText, описанной в другой работе Т. Миколова [28], ситуация несколько лучше, так как она рассчитана на работу с редкими словами. Но в силу того, что ее словарь также является ограниченным и зависит от обучающей выборки, то устойчивой к опечаткам ее также нельзя назвать. В рамках диссертационной работы была разработана программная система RoVe, которая может генерировать векторные представления для любой последовательности символов, что делает ее потенциально устойчивой к опечаткам. Для достижения лучшей устойчивости в этой системе используется контекст слова, для которого генерируется векторное представление. Как показали эксперименты, описанные в разделе 2.7, предложенная система является более устойчивой к шуму, нежели другие модели даже при помощи системы проверки орфографии.

Модели классификации текстов также, как правило, игнорируют существование шума в текстах. Существующие на сегодняшний день базовые модели классификации, такие как модели Кима из работы [99] и из работы [60]. Несмотря на то, что эти системы частично описаются на побуквенное представление слова, тем не менее явно устойчивость к шумам в них не была заложена, что было показано в экспериментах в разделе 3.3. Добавление модулей векторного представления слов RoVe и fastText делает описанные системы более устойчивыми к шуму.

На сегодняшний день общепризнанно лучшей системой для распознавания именованных сущностей на разных языках является модель biLSTM-CRF. Эта система показала лучшие результаты для английского языка в работе Лампля [102], для русского языка в работе Аня [101] и для французского языка в работе автора [12]. Несмотря на наличие в этой системе модуля для работы с побуквенным представлением слова, она не является устойчивой к шуму, что было

в показано в экспериментах раздела 4.4. Автором было предложено несколько расширений для этой системы, из которых расширение с использованием векторов fastText оказалось наиболее устойчивым к шумам в виде опечаток.

В задаче извлечения аспектов долгое время доминировали ненейросетевые подходы, в частности подход латентного размещения Дирихле, описанный в работе Бляя [68]. Эти подходы не являются устойчивыми к шуму в виде опечаток, так как полагаются на полное совпадение написания слова из текста написанию слова из словаря, что показали эксперименты в разделе 5.4. В 2017 году была представлена система ABAE, основанная на векторных представлениях слов, эта система была представлена в работе Хе и соавторов [69]. Эта система базируется на векторных представлениях слов Word2Vec, что делает ее уязвимой к вышеописанным проблемам графовых моделей. В работе автора [13] были представлены расширения этой системы, более устойчивые к шуму. В частности это были системы с использованием векторных представлений слов на основе побуквенного представления слова, а также векторных представлений из системы RoVe. Система RoVe в проведенных экспериментах показала лучшую устойчивость к шуму.

В ходе данной диссертационной работы были получены следующие результаты:

- Разработаны новые методы сравнения качества программных систем относительно их устойчивости к шуму для задач векторных представлений слов, классификации текстов, распознавания именованных сущностей и извлечения аспектов. Существующие аналоги разработанных методов применяются для оценки качества систем проверки орфографии и не предназначены для других задач. Также существенным отличием является наличие возможности регулирования уровня шума в разработанных методах.
- Разработаны новые методы построения программных систем устойчивых к шуму векторных представлений слов, классификации текстов и извлечения аспектов. Разработанные методы применены в описанных задачах и показали во многих экспериментах лучшие результаты.
- Создан, апробирован и внедрен программный комплекс, реализующий разработанные методы.

Полученные результаты могут быть применены в различных прикладных задачах обработки естественного языка, как в непосредственно рассмотренных,

а именно задаче классификации текста, извлечения именованных сущностей и извлечения аспектов, так и в других задачах, например, в задаче анализа тональности, применительно к конкретной сущности, задачам информационного поиска, распознавания речи и оптического распознавания текста, а также многим другим, где характерен шумные входные данные.

Дальнейшие перспективы развития исследований могут быть связаны с использованием предложенных моделей построения устойчивых векторных представлений в другим прикладным задачам обработки естественного языка, например, к задаче информационного поиска, суммаризации, прежде всего абстрактивной. Также перспективно применение описанных моделей к задаче машинного перевода в свете того, что Хайралла и Кён недавно переставили исследование характерных для задач машинного перевода типов шумов [19].

Стоит отметить, что разработанные программные комплексы по построению устойчивых векторных представлений, классификации текстов, распознавания именованных сущностей частично выложены в открытый доступ, что существенно упрощает дальнейшие исследования.

В заключение автор выражает благодарность и большую признательность научному руководителю Арлазарову В.Л. за поддержку, помощь, обсуждение результатов и научное руководство. Также автор благодарит Тутубалину Е.В. и Петренко А.К. за многочисленные замечания по тексту диссертации и тексту автореферата, Хахулина Т.А. и Лялина В.А. за их труд при постановке экспериментов, а также уважаемых оппонентов Тулупьева А.Л. и Иванова В.В. за их готовность выступить в таком непростом качестве. А также Лукашевич Н.В. за замечания по автореферату и презентации. Автор выражает признательность Логачевой В.К., Бурцеву М.С. и Браславскому П.И. за высказанные замечания по автореферату. Данная диссертационная работа была выполнена при поддержке Фонда поддержки проектов Национальной технологической инициативы и ПАО “Сбербанк”. Идентификатор проекта 0000000007417F630002.

Список сокращений и условных обозначений

w_i	вектор i-ого токена
h_i	вектор скрытого состояния на i-ом шаге (входном слове)
c_i	вектор состояния ячейки памяти на i-ом шаге (входном слове)
LSTM	Long Short-Term Memory, ячейка долговременной краткосрочной памяти
GRU	Gated Recurrent Unit, ячейка рекуррентной ячейки с гейтами
CNN	Convolutional Neural Network, сверточная нейронная сеть
RNN	Recurrent Neural Network, рекуррентная нейронная сеть
SRU	Simple Recurrent Unit, простая рекуррентная ячейка
OOV	Out of Vocabulary [word], несловарное слово
SVM	Support Vector Machine, машина опорных векторов
TF-IDF	Term Frequency, Inverced Document Frequency; частота термина, обратная документная частота
CRF	Conditional Random Field, условное случайное поле
LDA	Latent Dirichlet Allocation, латентное размещение Дирихле
ROC-AUC	Receiver Operating Characteristic Area Under the Curve, площадь под кривой ROC
TPR	True Positive Rate, доля предсказанных верно
FPR	False Positive Rate, доля предсказанных неверно

Словарь терминов

Векторные представления (embedding) - сопоставление некоторому объекту (например, слову, предложению) вектора в n-мерном пространстве.

OOV - слово, не содержащееся в словаре.

One-hot - способ векторного представления слов, где для каждого слова создается вектор, длиной соответствующей количеству слов в словаре, со всеми нулями, кроме одной единицы, которая ставится под номером слова в словаре. Аналогично термин может применять к символам, тогда в качестве словаря выступает алфавит, в остальном описание сохраняет силу.

Аспект – мнение по конкретным свойствам или частям сущности [111]. Например, Обзор ресторана одновременно может описывать несколько аспектов, таких, как еда, обслуживание, атмосфера в ресторане.

Контекст – последовательность слов, как правило является целиком предложением или его частью. Выделяют *правый* и *левый* контексты, которые являются частью общего контекста и находятся соответственно справа и слева от центрального слова.

Подслово - часть слова, морфема или n-грамма.

Токен (token, также term) - единица обработки текста, в самом простом случае - слово.

Эмоджи - текстовое представление эмоции или ассоциированного с эмоцией действия, например, эмоджи ":)" является условным обозначением улыбки, т.е. позитивной эмоции.

F_1-мера - мера качества в задаче классификации, определяемая как

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

где *precision* - это доля угаданных верно, среди предсказанных объектов целевого класса, а *recall* - доля предсказанных верно среди всех объектов целевого класса. F_1 для многоклассовой классификации считается независимо для каждого класса в режиме "целевой класс - все остальные классы". Существует несколько стратегий получения метрики качества F_1 для многоклассовой классификации, а именно - подсчет сначала бинарной метрики F_1 для каждого класса и усреднение полученных значений, это так называемое *макро-усреднение*; разновидность макро-усреднения, где при усреднении учитывается, какую

долю он составляет с датасете, называется “взвешенное по классам” усреднение; подсчет метрики F_1 для всех классов одновременно - “микро-усреднение”.

SoftMax - функция “мягкого” максимума, мягкого в смысле дифференцируемости. Была предложена в работе [112]. Представляет собой обобщение логистической регрессии. Формально записывается следующим образом:

$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}},$$

где $x_1, .., x_n$ - произвольный набор значений.

Говоря, что “функция SoftMax выдает распределение” автор имеет в виду, что функция SoftMax была применена ко всем значениям из набора и далее работа идет с набором значений функции. Распределением в формальном смысле теории вероятностей полученный набор значений будет только в случае, если изначальный набор значений насэмплирован из нормального распределения. Но на практике этим требованием часто пренебрегают.

Документ - набор последовательных токенов. В этом смысле документов может являться, как новостная статья, так и отдельное предложение.

Корпус - набор документов.

Подкорпус - подмножество множества документов корпуса.

SVM - метод линейной классификации, основанный на поиске оптимальной разделяющей гиперплоскости в пространстве признаков.

Словоформа - вариант написания слова; может быть нормативной, когда такой вариант написания слова допускается грамматикой языка (например, форма родительного падежа единственного числа для существительного), а может быть шумовой в обратном случае.

Лемматизация - процесс приведения слова к начальной форме, т.е. каждой словоформе сопоставляется специально выбранная словоформа, например, для глаголов - это форма инфинитива.

Батч или **мини-батч** - подвыборка в процессе обучения или тестирования модели машинного обучения. Операции над объектами из мини-батча выполняются одновременно, то есть параллельно.

ROC-AUC - мера качества для бинарной классификации, основанная на построении графика соотношения количества объектов тестовой выборки предсказанных верно для позитивного класса (True Positive Rate, TPR) и количества объектов, ошибочно отнесенных к позитивному классу (False Posistive

Rate, FPR). В качестве меры качества берется площадь под построенным графиком, называемым ROC-кривой. Пример ROC-кривой можно посмотреть на рисунке 6.1.

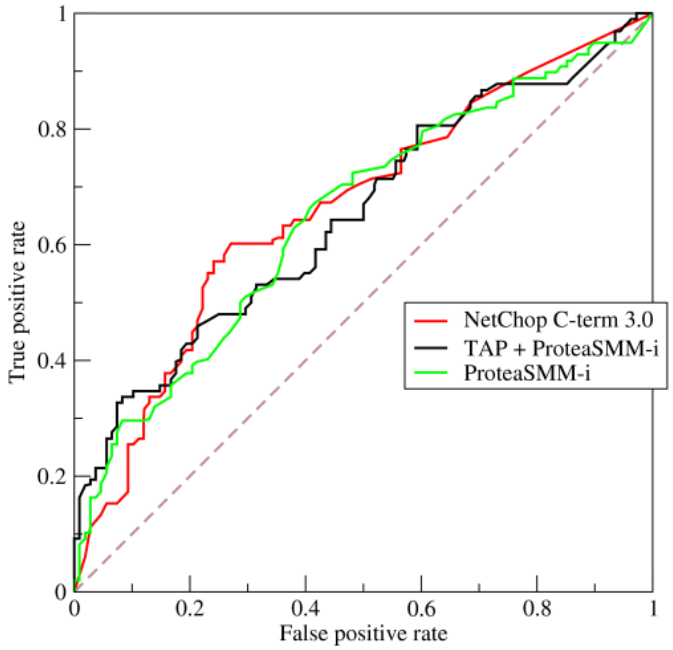

Рисунок 6.1 — Пример построения ROC-кривых для бинарной классификации. Иллюстрация взята из Wikipedia.

TPR - мера качества бинарной классификации, определяемая, как соотношение верно предсказанных объектов позитивного класса (True Positive) ко всем объектам позитивного класса (Positive): $TPR = \frac{TruePositive}{Positive}$

FPR - мера качества бинарной классификации, определяемая, как соотношение ложно предсказанных объектов позитивного класса (False Positive) ко всем объектам негативного класса (Negative): $FPR = \frac{FalsePositive}{Negative}$

Список литературы

1. *Liu, B.* Scalable sentiment classification for big data analysis using naive bayes classifier / B. Liu [и др.] // Big Data, 2013 IEEE International Conference on. — IEEE. 2013. — С. 99—104.
2. *Cucerzan, S.* Spelling correction as an iterative process that exploits the collective knowledge of web users / S. Cucerzan, E. Brill // Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. — 2004.
3. *Левенштейн, В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов / В. И. Левенштейн // Доклады Академии наук. Т. 163. — Российская академия наук. 1965. — С. 845—848.
4. *Sorokin, A.* Spelling correction for morphologically rich language: a case study of Russian / A. Sorokin // Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing. — 2017. — С. 45—53.
5. *Malykh, V.* Robust to Noise Context-Aware Word Vectors / V. Malykh, V. Logacheva, T. Khakhulin // Записки научных семинаров ПОМИ. Серия “искусственный интеллект”. — 2019.
6. *Mikolov, T.* Distributed representations of words and phrases and their compositionality / T. Mikolov [и др.] // Advances in neural information processing systems. — 2013. — С. 3111—3119.
7. *Niu, J.* Multi-task Character-Level Attentional Networks for Medical Concept Normalization / J. Niu [и др.] // Neural Processing Letters. — 2018. — С. 1—18.
8. *Belinkov, Y.* Synthetic and natural noise both break neural machine translation / Y. Belinkov, Y. Bisk // Proceedings of Seventh International Conference on Learning Representations. — 2018.
9. *Малых, В.* К вопросу о классификации шумных текстов / В. Малых, В. Лялин // Труды ИСА РАН. Специальный выпуск. — 2018.
10. *Malykh, V.* Reproducing Russian NER Baseline Quality without Additional Data. / V. Malykh, A. Ozerin // CDUD at CLA. — 2016. — С. 54—59.

11. *Malykh, V.* Robust word vectors for Russian language / V. Malykh // Proceedings of Artificial Intelligence and Natural Language AINL FRUCT 2016 Conference, Saint-Petersburg, Russia. — 2016. — С. 10—12.

12. *Malykh, V.* Named Entity Recognition in Noisy Domains / V. Malykh, V. Lyalin // The Proceedings of the 2018 International Conference on Artificial Intelligence: Applications and Innovations. — 2018.

13. *Malykh, V.* Noise Robustness in Aspect Extraction Task / V. Malykh, T. Khakhulin // The Proceedings of the 2018 Ivannikov ISP RAS Open Conference. — 2018.

14. *Malykh, V.* Improving Classification Robustness for Noisy Texts with Robust Word Vectors / V. Malykh, V. Lyalin // Записки научных семинаров ПОМИ. Серия “искусственный интеллект”. — 2019.

15. *Malykh, V.* What Did You Say? On Classification of Noisy Texts / V. Malykh, V. Lyalin // XX Международная научно-техническая конференция “Нейроинформатика-2018’: Сборник научных трудов. В 2-х частях. Ч. 1. — М. : НИЯУ МИФИ, 2018.

16. *Carlson, A.* Memory-based context-sensitive spelling correction at web scale / A. Carlson, I. Fette // Machine learning and applications, 2007. ICMLA 2007. sixth international conference on. — IEEE. 2007. — С. 166—171.

17. *Islam, A.* Real-word spelling correction using Google Web IT 3-grams / A. Islam, D. Inkpen // Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3. — Association for Computational Linguistics. 2009. — С. 1241—1249.

18. *Xie, Z.* Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction / Z. Xie [и др.] // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Т. 1. — 2018. — С. 619—628.

19. *Khayrallah, H.* On the Impact of Various Types of Noise on Neural Machine Translation / H. Khayrallah, P. Koehn // Proceedings of the 2nd Workshop on Neural Machine Translation and Generation. — 2018. — С. 74—83.

20. *Heigold, G.* How Robust Are Character-Based Word Embeddings in Tagging and MT Against Wrod Scramlbing or Randdm Nouse? / G. Heigold [и др.]. — 2018.

21. *Salton, G.* Term-weighting approaches in automatic text retrieval / G. Salton, C. Buckley // Information processing & management. — 1988. — Т. 24, № 5. — С. 513—523.

22. *Rong, X.* word2vec Parameter Learning Explained / X. Rong. — 2014. — Нояб.

23. *Mikolov, T.* Subword language modeling with neural networks / T. Mikolov [и др.]. — 2012. — Февр.

24. *Huang, E. H.* Improving word representations via global context and multiple word prototypes / E. H. Huang [и др.] // Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. — Association for Computational Linguistics. 2012. — С. 873—882.

25. *Joulin, A.* Bag of tricks for efficient text classification / A. Joulin [и др.] // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. 2, Short Papers. — 2017. — С. 427—431.

26. *Smith, N. A.* Contrastive estimation: Training log-linear models on unlabeled data / N. A. Smith, J. Eisner // Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. — Association for Computational Linguistics. 2005. — С. 354—362.

27. *Firth, J. R.* A synopsis of linguistic theory, 1930-1955 / J. R. Firth // Studies in linguistic analysis. — 1957.

28. *Bojanowski, P.* Enriching word vectors with subword information / P. Bojanowski [и др.] // arXiv preprint arXiv:1607.04606. — 2016.

29. *Bochkarev, V. V.* The average word length dynamics as an indicator of cultural changes in society / V. V. Bochkarev, A. V. Shevlyakova, V. D. Solovyev // Social Evolution & History. — 2015. — Т. 14, № 2. — С. 153—175.

30. *Pennington, J.* Glove: Global vectors for word representation / J. Pennington, R. Socher, C. Manning // Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). — 2014. — С. 1532—1543.

31. *Ling, W.* Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation / W. Ling [и др.]. — 2015.

32. *Pinter, Y.* Mimicking Word Embeddings using Subword RNNs / Y. Pinter, R. Guthrie, J. Eisenstein // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. — 2017. — С. 102—112.

33. *Astudillo, R.* Learning Word Representations from Scarce and Noisy Data with Embedding Subspaces / R. Astudillo [и др.] // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). — Beijing, China : Association for Computational Linguistics, 2015. — С. 1074—1084. — URL: http://www.aclweb.org/anthology/P15-1104.

34. *Nguyen, K. A.* Neural-based Noise Filtering from Word Embeddings / K. A. Nguyen, S. Schulte im Walde, N. T. Vu. — 2016. — URL: http://www.aclweb.org/anthology/C16-1254.

35. *Vylomova, E.* Word Representation Models for Morphologically Rich Languages in Neural Machine Translation / E. Vylomova [и др.] // Proceedings of the First Workshop on Subword and Character Level Models in NLP. — 2017. — С. 103—108.

36. *Zhang, X.* Character-level convolutional networks for text classification / X. Zhang, J. Zhao, Y. LeCun // Advances in Neural Information Processing Systems. — 2015. — С. 649—657.

37. *Saxe, J.* eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys / J. Saxe, K. Berlin // CoRR. — 2017. — Т. abs/1702.08568. — arXiv: 1702.08568. — URL: http://arxiv.org/abs/1702.08568.

38. *Wehrmann, J.* A character-based convolutional neural network for language-agnostic twitter sentiment analysis / J. Wehrmann [и др.] // IJCNN-2017: International Joint Conference on Neural Networks. — 2017. — С. 2384—2391.

39. *Kiela, D.* Context-Attentive Embeddings for Improved Sentence Representations / D. Kiela, C. Wang, K. Cho // CoRR. — 2018. — Т. abs/1804.07983. — arXiv: 1804.07983. — URL: http://arxiv.org/abs/1804.07983.

40. *McCann, B.* Learned in translation: Contextualized word vectors / B. McCann [и др.] // Advances in Neural Information Processing Systems. — 2017. — С. 6294—6305.

41. *Peters, M. E.* Deep contextualized word representations / M. E. Peters [и др.] // Proceedings of NAACL-HLT 2018. — 2018. — С. 2227—2237.

42. *Nielsen, M. A.* Neural networks and deep learning. Т. 25 / M. A. Nielsen. — Determination press USA, 2015.

43. *Goodfellow, I.* Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. — MIT Press, 2016. — http://www.deeplearningbook.org.

44. *Bengio, Y.* Representation Learning: A Review and New Perspectives / Y. Bengio, A. Courville, P. Vincent // IEEE Trans. Pattern Anal. Mach. Intell. — Washington, DC, USA, 2013. — Авг. — Т. 35, № 8. — С. 1798—1828. — URL: http://dx.doi.org/10.1109/TPAMI.2013.50.

45. *Collobert, R.* Natural language processing (almost) from scratch / R. Collobert [и др.] // Journal of Machine Learning Research. — 2011. — Т. 12, Aug. — С. 2493—2537.

46. *Vaswani, A.* Attention is all you need / A. Vaswani [и др.] // Advances in Neural Information Processing Systems. — 2017. — С. 5998—6008.

47. *Nair, V.* Rectified linear units improve restricted boltzmann machines / V. Nair, G. E. Hinton // Proceedings of the 27th international conference on machine learning (ICML-10). — 2010. — С. 807—814.

48. *Montavon, G.* Tricks of the Trade / G. Montavon, G. B. Orr, K.-R. Müller. — 1998.

49. *Srivastava, N.* Dropout: a simple way to prevent neural networks from overfitting / N. Srivastava [и др.] // The Journal of Machine Learning Research. — 2014. — Т. 15, № 1. — С. 1929—1958.

50. *Pineda, F. J.* Generalization of back-propagation to recurrent neural networks / F. J. Pineda // Physical review letters. — 1987. — Т. 59, № 19. — С. 2229.

51. *Cho, K.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation / K. Cho [и др.] // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — 2014. — С. 1724—1734.

52. *Bengio, Y.* Learning long-term dependencies with gradient descent is difficult / Y. Bengio, P. Simard, P. Frasconi // IEEE Transactions on Neural Networks. — 1994. — Март. — Т. 5, № 2. — С. 157—166.

53. *Schuster, M.* Bidirectional recurrent neural networks / M. Schuster, K. K. Paliwal // IEEE Transactions on Signal Processing. — 1997. — Т. 45, № 11. — С. 2673—2681.

54. *Hochreiter, S.* Long Short-Term Memory / S. Hochreiter, J. Schmidhuber // Neural Computation. — 1997. — Т. 9, № 8. — С. 1735—1780. — Based on TR FKI-207-95, TUM (1995).

55. *Gers, F. A.* Recurrent nets that time and count / F. A. Gers, J. Schmidhuber // Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on. Т. 3. — IEEE. 2000. — С. 189—194.

56. *Johnson, R.* Supervised and semi-supervised text categorization using LSTM for region embeddings / R. Johnson, T. Zhang // Proceedings of the 33rd International Conference on International Conference on Machine Learning. Т. 48. — JMLR. org. 2016. — С. 526—534.

57. *Chung, J.* Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling / J. Chung [и др.] // NIPS 2014 Workshop on Deep Learning, December 2014. — 2014.

58. *Bahdanau, D.* Neural machine translation by jointly learning to align and translate / D. Bahdanau, K. Cho, Y. Bengio // Proceedings of Fourth International Conference on Learning Representations. — 2015.

59. *Fukushima, K.* Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position / K. Fukushima, S. Miyake // Pattern recognition. — 1982. — Т. 15, № 6. — С. 455—469.

60. *Kim, Y.* Character-Aware Neural Language Models. / Y. Kim [и др.] // AAAI. — 2016. — С. 2741—2749.

61. *Vinciarelli, A.* Noisy text categorization / A. Vinciarelli // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2005. — Т. 27, № 12. — С. 1882—1895.

62. *Li, Y.* Learning robust representations of text / Y. Li, T. Cohn, T. Baldwin // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. — 2016. — С. 1979—1985.

63. *Tutubalina, E.* Inferring Sentiment-Based Priors in Topic Models / E. Tutubalina, S. I. Nikolenko // Proc. 14th Mexican International Conference on Artificial Intelligence. Т. 9414. — Springer, 2015. — С. 92—104. — (Lecture Notes in Computer Science).

64. *Lafferty, J. D.* Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data / J. D. Lafferty, A. McCallum, F. C. N. Pereira // Proceedings of the Eighteenth International Conference on Machine Learning. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. — С. 282—289. — (ICML '01). — URL: http://dl.acm.org/citation.cfm?id=645530.655813.

65. *Moon, S.* Multimodal Named Entity Disambiguation for Noisy Social Media Posts / S. Moon, L. Neves, V. Carvalho // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Т. 1. — 2018. — С. 2000—2008.

66. *Papadimitriou, C. H.* Latent semantic indexing: A probabilistic analysis / C. H. Papadimitriou [и др.] // Journal of Computer and System Sciences. — 2000. — Т. 61, № 2. — С. 217—235.

67. *Hofmann, T.* Probabilistic Latent Semantic Indexing / T. Hofmann // Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. — Berkeley, California, USA : ACM, 1999. — С. 50—57. — (SIGIR '99). — URL: http://doi.acm.org/10.1145/312624.312649.

68. *Blei, D. M.* Latent dirichlet allocation / D. M. Blei, A. Y. Ng, M. I. Jordan // Journal of machine Learning research. — 2003. — Т. 3, Jan. — С. 993—1022.

69. *He, R.* An unsupervised neural attention model for aspect extraction / R. He [и др.] // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Т. 1. — 2017. — С. 388—397.

70. *Veselovská, K.* ÚFAL: Using hand-crafted rules in aspect based sentiment analysis on parsed data / K. Veselovská, A. Tamchyna // Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). — 2014. — С. 694—698.

71. *Wang, H.* Product weakness finder: an opinion-aware system through sentiment analysis / H. Wang, W. Wang // Industrial Management & Data Systems. — 2014. — Т. 114, № 8. — С. 1301—1320.

72. *Pablos, A. G.* Unsupervised acquisition of domain aspect terms for Aspect Based Opinion Mining / A. G. Pablos [и др.] // Procesamiento del Lenguaje Natural. — 2014. — Т. 53. — С. 121—128.

73. *S. Harris, Z.* Distributional Structure / Z. S. Harris // Word. — 1954. — Авг. — Т. 10. — С. 146—162.

74. *Bengio, Y.* A Neural Probabilistic Language Model / Y. Bengio, R. Ducharme, P. Vincent. — 2000. — Янв.

75. *Botha, J. A.* Compositional Morphology for Word Representations and Language Modelling / J. A. Botha, P. Blunsom. — 2014. — Май.

76. *Ling, W.* Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation / W. Ling [и др.]. — 2015. — URL: http://www.aclweb.org/anthology/D15-1176.

77. *Mikolov, T.* Extensions of recurrent neural network language model / T. Mikolov [и др.] // Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. — IEEE. 2011. — С. 5528—5531.

78. *Koutnik, J.* A Clockwork RNN / J. Koutnik [и др.] // Proceedings of the 31st International Conference on Machine Learning. — 2014. — С. 1863—1871.

79. *Lei, T.* Training RNNs as Fast as CNNs / T. Lei, Y. Zhang. — 2017. — Сент.

80. *Sakaguchi, K.* Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network. / K. Sakaguchi [и др.] // AAAI. — 2017. — С. 3281—3287.

81. *Seo, M.* Bidirectional Attention Flow for Machine Comprehension / M. Seo [и др.]. — 2016. — Нояб.

82. *Kingma, D.* Adam: A Method for Stochastic Optimization / D. Kingma, J. Ba. — 2014. — Дек.

83. *Dolan, B.* Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources / B. Dolan, C. Quirk, C. Brockett. — 2004. — Янв.

84. *Bowman, S.* A large annotated corpus for learning natural language inference / S. Bowman [и др.]. — 2015. — Авг.

85. *Socher, R.* Recursive deep models for semantic compositionality over a sentiment treebank / R. Socher [и др.]. — 2013. — Янв.

86. *Pronoza, E.* Construction of a Russian paraphrase corpus: unsupervised paraphrase extraction / E. Pronoza, E. Yagunova, A. Pronoza // Russian Summer School in Information Retrieval. — Springer. 2015. — С. 146—157.

87. *Rubtsova, Y.* Automatic Term Extraction for Sentiment Classification of Dynamically Updated Text Collections into Three Classes / Y. Rubtsova // International Conference on Knowledge Engineering and the Semantic Web. — Springer. 2014. — С. 140—149.

88. *Demir, S.* Turkish Paraphrase Corpus. / S. Demir [и др.] // LREC. — Citeseer. — С. 4087—4091.

89. *Лукашевич, Н.* SentiRuEval: тестирование систем анализа тональности текстов на русском языке по отношению к заданному объекту / Н. Лукашевич [и др.] // Компьютерная лингвистика. М. — 2015. — С. 13.

90. *Polikarpov.* Towards the Foundations of Menzerath's Law / Polikarpov. — 2007. — Нояб.

91. *Lewis, D. D.* Rcv1: A new benchmark collection for text categorization research / D. D. Lewis [и др.] // Journal of machine learning research. — 2004. — Т. 5, Apr. — С. 361—397.

92. *Андрющенко, В. М.* Концепция и архитектура машинного фонда русского языка / В. М. Андрющенко. — Наука, 1989.

93. *Kutuzov, A.* WebVectors: a toolkit for building web interfaces for vector semantic models / A. Kutuzov, E. Kuzmenko // International Conference on Analysis of Images, Social Networks and Texts. — Springer. 2016. — С. 155—161.

94. *Segalovich, I.* A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. / I. Segalovich // MLMTA. — Citeseer. 2003. — С. 273—280.

95. *Yildirim, O.* 42 Bin Haber Veri Kumesi / O. Yildirim, F. Atik, M. F. Amasyali // Yildiz Teknik Universitesi, Bilgisayar Muh. Bolumu. — 2003.

96. *Porter, M. F.* Snowball: A language for stemming algorithms / M. F. Porter. — 2001.

97. *Grave, E.* Bag of tricks for efficient text classification / E. Grave [и др.] // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL. — 2017. — С. 3—7.

98. *Howard, J.* Universal language model fine-tuning for text classification / J. Howard, S. Ruder // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Т. 1. — 2018. — С. 328—339.

99. *Kim, Y.* Convolutional neural networks for sentence classification / Y. Kim // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — 2014. — С. 1746—1751.

100. *Glorot, X.* Understanding the difficulty of training deep feedforward neural networks / X. Glorot, Y. Bengio // Proceedings of the thirteenth international conference on artificial intelligence and statistics. — 2010. — С. 249—256.

101. *Le, T. A.* Application of a Hybrid Bi-LSTM-CRF Model to the Task of Russian Named Entity Recognition / T. A. Le, M. Y. Arkhipov, M. S. Burtsev // Conference on Artificial Intelligence and Natural Language. — Springer. 2017. — С. 91—103.

102. *Lample, G.* Neural Architectures for Named Entity Recognition / G. Lample [и др.] // Proceedings of NAACL-HLT. — 2016. — С. 260—270.

103. *Tjong Kim Sang, E. F.* Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition / E. F. Tjong Kim Sang, F. De Meulder // Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. — Association for Computational Linguistics. 2003. — С. 142—147.

104. *Власова, Н.* Сообщение о русскоязычной коллекции для задачи извлечения личных имен из текстов / Н. Власова, Е. Сулейманова, И. Трофимов // Труды конференции по компьютерной и когнитивной лингвистике TEL'2014 "Языковая семантика: модели и технологии". — 2014. — С. 36—40.

105. *Mozharova, V.* Two-stage approach in Russian named entity recognition / V. Mozharova, N. Loukachevitch // Intelligence, Social Media and Web (ISMW FRUCT), 2016 International FRUCT Conference on. — IEEE. 2016. — С. 1—6.

106. *Cédric, L.* CAp 2017 challenge: Twitter Named Entity Recognition / L. Cédric [и др.]. — 2017. — Июль.

107. *Weston, J.* Wsabie: Scaling up to large vocabulary image annotation / J. Weston, S. Bengio, N. Usunier // IJCAI. Т. 11. — 2011. — С. 2764—2770.

108. *Li, P.* Enhancing Sentence Relation Modeling with Auxiliary Character-level Embedding / P. Li, H. Huang. — 2016. — Март.

109. *Ganu, G.* Beyond the stars: improving rating predictions using review text content. / G. Ganu, N. Elhadad, A. Marian // WebDB. Т. 9. — Citeseer. 2009. — С. 1—6.

110. *Intel.* Intel® 64 and IA-32 Architectures Software Developer's Manual. Т. 2 / Intel. — 2016. — URL: https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf.

111. *Большакова, Е.* Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учеб. пособие / Е. Большакова [и др.] // М.: МИЭМ. — 2011. — Т. 272. — С. 3.

112. *McCullagh, P.* Generalized linear models. Т. 37 / P. McCullagh, J. A. Nelder. — CRC press, 1989.

Список рисунков

Список таблиц