Правительство Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский университет

«Высшая школа экономики»

Факультет гуманитарных наук

**Образовательная программа**

**«Фундаментальная и компьютерная лингвистика»**

**КУРСОВАЯ РАБОТА**

На тему: Автоматическое обнаружение слов русского языка с короткой историей (на материале газетных текстов последнего десятилетия).

*Тема на английском: Automatic recognition of Russian words with a short history (on a newspaper texts of a passed 15 years)*

Студентка 3 курса

группы № 162

Пичужкина Ольга Валерьевна

(Ф.И.О.)

Научный руководитель

Орехов Борис Валерьевич

(Ф.И.О)

к. ф. н., доцент Школы лингвистики факультета гуманитарных наук НИУ ВШЭ

(должность, звание)

Москва, 2018 г.

Оглавление

1. Введение…………………………………………………………………………………...
2. Этапы работы……………………………………………………………………………...
3. Результаты…………………………………………………………………………………
4. Заключение………………………………………………………………………………...
5. Литература…………………………………………………………………………………
6. Приложение………………………………………………………………………………..

*“Судьба слов далеко не так безоблачна, как кажется на первый взгляд. Среди множества новых слов, появляющихся в последнее время в русских текстах, лишь некоторым удается закрепиться в языке надолго или даже остаться в нем. Другие же напоминают незваных гостей, которые, потоптавшись в передней, вскоре незаметно покидают отвергнувший их дом”.*

*М. А. Кронгауз, “Русский язык на грани нервного срыва”.*

1. **Введение**

Слова с короткой историей – слова русского языка, появившиеся в информационном поле в связи с каким-либо общественно-значимым событием, а затем утратившие актуальность и забывшиеся. К таким окказионализмам могут относиться, например, слова, уникальные для речи какого-либо политика, потерявшие актуальность после ухода политика с политической сцены (*консенсус*), слова, потерявшие актуальность, потому что перестало существовать явление, ими обозначаемое (*пейджер, милиция*), это может быть придуманный кем-то не прижившийся в языке окказионализм (*сетература*), и т. д.

История таких окказионализмов хорошо отслеживается по новостным текстам, сохранившимся в сети Интернет. При наличии достаточно большого объёма данных можно попытаться находить такие слова автоматически. Задача данного исследования – разработка алгоритма, автоматически обнаруживающего в достаточно большом корпусе газетных текстов (полностью выкачанные сайты нескольких интернет-СМИ, содержащие в себе тексты, написанные в период с 2000 года до 2018 года) слова с короткой историей. Анализ окказионализмов не предполагается.

1. **Этапы работы.**

Код программ написан на языке Python 3. Этот язык программирования очень удобен при работе с естественным языком. Для удобства и наглядности код всех программ предоставлен не в формате py , а в формате Jupyter Notebook (ipynb). Это формат, позволяющий запускать ячейки кода отдельно, что может быть удобно для визуализации данных.

Работа проводилась в три этапа: сбор данных, составление инвертированного индекса по датам для корпуса газетных текстов и обнаружение слов с короткой историей по инвертированному индексу.

Был собран газетный корпус объёмом 1257850 текстов. Источниками стали четыре новостных Интернет-СМИ – Lenta.ru (<https://lenta.ru>, 734019 текстов), Полит.ру (<https://polit.ru/>, 323919 текстов), Znak.com (<https://www.znak.com>, 42225 текстов), и сайт информационного агентства Интерфакс (<https://www.interfax.ru>, 157687 текстов). Эти СМИ были выбраны, поскольку они:

* + Индексируемые (соответственно, легко выкачать все тексты за определённый промежуток времени с сохранением даты).
  + Нетематические (соответственно, корпус текстов будет репрезентативен).
  + Достаточно объёмные.

Lenta.ru – одно из главных российских интернет-изданий, основанное в 1999 году Антоном Носиком. Polit.ru – общественно-политическое российское интернет-издание, основанное 21 февраля 1998 года. Znak.com – российское интернет-СМИ, основанное 3 декабря 2012 года журналисткой Аксаной Пановой. Интерфакс – независимое российское информационное агентство, появившееся в сентябре 1989 года.

Выкачивались все тексты в период с 1 января 2000 года до 31 декабря 2018 года включительно, для каждого текста сохранялась дата написания (как объект datetime.date, в формате ГГГГ-ММ-ДД). Для текстов издания Znak.com также сохранялся автор статьи (так как другие три интернет-издания не указывают автора текстов). При сборе данных использовались библиотеки pandas (библиотека для работы с большими объёмами данных), BeautifulSoup (библиотека для парсинга html-страниц, позволяющая доставать из html-страницы такую информацию, как список ссылок на статьи, текст статьи, дату написания текста, автора статьи, если таковой есть, и т. д.), pickle (библиотека для сохранения объектов Python как файлы и загрузки из файла объектов Python, которая использовалась для того, чтобы сохранять и загружать корпус и инвертированный индекс по корпусу), http и urllib (библиотеки для работы со ссылками и html-страницами – urllib использовалась для выкачивания страниц, а http для обработки ошибок при выкачивани).

Для каждого из четырёх СМИ краулер по новостным сайтам выкачивал список ссылок на статьи за каждый день с 1 января 2000 года до 31 декабря 2018 года. Затем он проходил по каждой из этих статей и сохранял текст, дату написания и, если известно, автора текста (только для текстов со Znak.com). Полученный корпус затем был преобразован в формат датафрейма pandas и был сохранён как файл в формате pkl (объект Python, который можно открыть с помощью библиотеки pickle). Объём корпуса не позволял сохранить его в формате csv, не повредив данные.

Затем для корпуса был составлен инвертированный индекс. Инвертированный индекс (англ. *inverted index*) – структура данных, в которой для каждого слова текстового корпуса сохранён список документов корпуса, в которых оно встречается. Обычно инвертированный индекс используется для задачи поиска по текстам. Так как для нашей задачи имеют значение только даты, в которые встречается слово, для каждого слова сохранялся не список документов, в которых оно встречается, а список дат написания каждого из таких документов.

Перед составлением инвертированного индекса тексты корпуса были лемматизированы – все слова в них были приведены к начальной форме и нижнему регистру, пунктуация удалена. Для лемматизации использовалась библиотека pymorphy2[fast], выбранная из-за существенной (в сравнении с аналогичной библиотекой pymystem3) быстроты работы. Pymorphy2 – это морфологический анализатор для русского языка, написанный с использованием словаря OpenCorpora на языке Python, распространяющийся под лицензией MIT и предназначенный для работы под языками Python 2 и Python 3 [Korobov, 2015]. Он умеет обрабатывать незнакомые слова, поэтому подходит для нашей задачи.

При составлении инвертированного индекса не учитывались служебные слова (список служебных слов для русского был получен с помощью библиотеки NLTK, Natural Language Toolkit – самой большой библиотеки для обработки естественной речи), самые частотные и числительные. Кроме того, не учитывались “слова”, не содержавшие в себе ни одного буквенного символа (в том числе пустые «слова») – такие могли появиться случайно в процессе токенизации. Список 1000 самых частотных лемм для русского языка, которые не учитывались при составлении инвертированного индекса, был получен с помощью csv-версии частотного словаря современного русского языка О. Н. Ляшевской и С. А. Шарова [Ляшевская, Шаров, 2009]. Это частотный словарь современного русского языка, созданный при помощи Национального корпуса русского языка (<http://www.ruscorpora.ru> ), доступный онлайн в форматах html, pdf и csv (мы использовали формат csv) и содержащий в себе список упорядоченных по частотности лемм современного русского языка. Также не учитывались именованные сущности (потому что в противном случае в список слов с короткой историей попало бы значительное количество именованных сущностей).

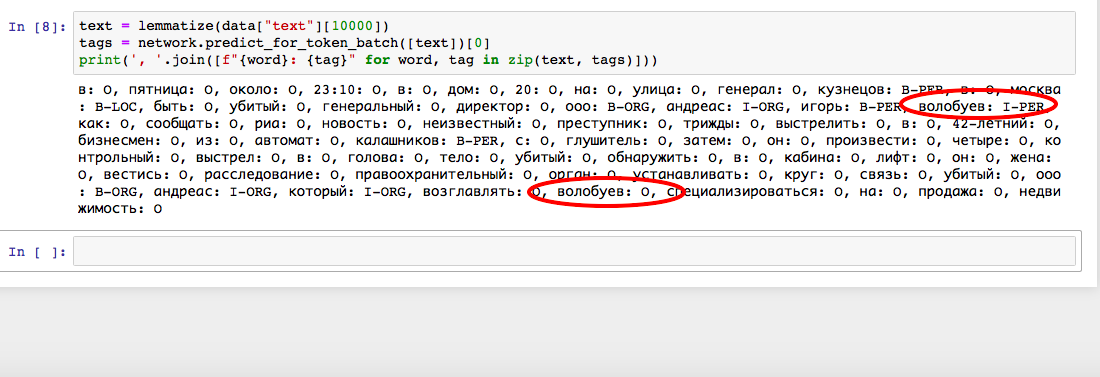
Для распознавания именованных сущностей использовалась библиотека NER (<https://github.com/deepmipt/ner>), распространяющаяся под лицензией Apache License 2.0. Библиотека включает в себя Bi-LSTM-CRF нейронную сеть (архитектура сети подробно описана в [Anh et al. 2017]) для распознавания именованных сущностей в русском, обученную на трёх корпусах – датасете из [Gareev et al. 2013], корпусе factRuEval-2016 (<https://github.com/dialogue-evaluation/factRuEval-2016>, [Starostin et al. 2016]) и корпусе Named Entities 3 (<http://labinform.ru/pub/named_entities/descr_ne.htm>, [Можарова, Лукашевич 2016]). Все три корпуса представляют собой большие коллекции текстов с размеченными вручную именованными сущностями. Нейронная сеть может распознавать такие именованные сущности, как людей (тэг *PER*), организации (тэг *ORG*) и локации (тэг *LOC*). F1-мера качества модели[[1]](#footnote-1), обученной на каждом из трёх корпусов по отдельности, составляет 87.17, 99.26 и 82.10 соответственно, что очень высоко. Библиотека позволяет использовать уже обученную нейронную сеть, а также позволяет обучить свою, в данной работе использовалась уже обученная.

Составленный инвертированный индекс (в виде словаря Python) был сохранён в формате pkl (объект Python, который можно открыть с помощью библиотеки pickle). Всего в инвертированном индексе содержится 481450 лемм слов.

Третий этап состоял в попытке автоматически обнаружить в газетном корпусе слова с короткой историей. Для этого было применено “сглаживание” – для каждого слова подсчитывалось количество вхождений в некотором временном окне (длиной 1 месяц) и сравнивалось с ожидаемым. Неологизмами считались те слова, у которых было 0 вхождений за все месяцы 2000, 2001 и 2018 годов (т. е., слова, появившиеся в корпусе не раньше 2002 года и исчезнувшие из корпуса не позже 2017 года). Исключались слова, у которых было менее пяти вхождений в корпус (т. к., небольшое количество вхождений ничего не говорит об истории слова и не даёт построить график, кроме того, слова с маленьким количеством вхождений вполне могут быть несуществующими и попавшими в корпус благодаря ошибкам или опечаткам). Сглаживание также использовалось для построения графиков слов (см. ниже) – график вхождений слова, учитывающий количество вхождений за каждый день, был бы слишком “рваным” и малоинформативным, поэтому при построении графиков учитывалось количество вхождений за каждый месяц. Для сглаживания была написана функция, получающая на вход слово и возвращающая список длиной 228 (так как между 01.01.2000 и 01.01.2019 всего прошло 19 \* 12 = 228 месяцев), в котором i-ая позиция – количество вхождений слова за i-ый месяц (начиная с января 2000 года).

В процессе работы обнаружилось, что в газетных текстах достаточно много слов с опечатками, что привело к тому, что в инвертированном индексе достаточно много “лишних” слов. Возможно, перед подсчётом инвертированного индекса стоило удалять не только стоп-слова, частотные слова и именованные сущности, но и слова с опечатками. Так как у таких слов, как правило, одно (в редких случаях – два) вхождения в корпусе, при поиске слов с короткой историей они не учитывались. Кроме того, нейронное распознавание именованных сущностей иногда даёт ложноотрицательный результат. Это видно, например, из примера ниже, где встретившееся дважды в одном и том же тексте слово “Волобуев” размечено в первом случае как именованная сущность (персона), а во втором случае не как именованная сущность:





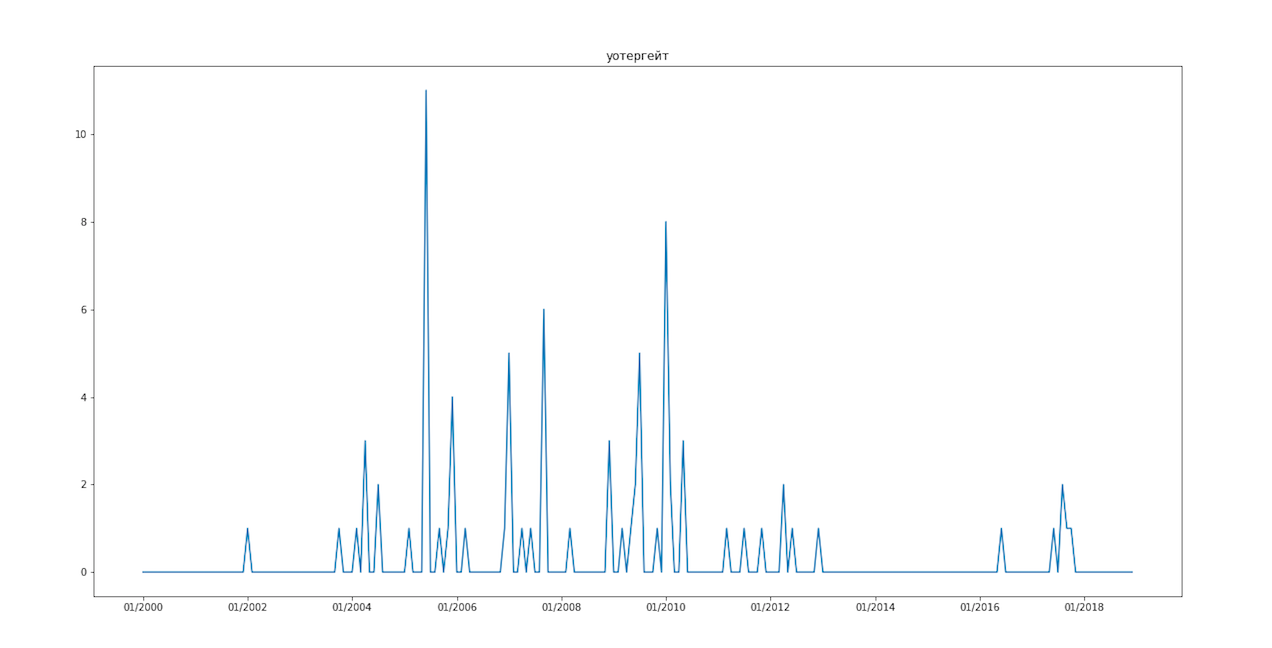
*Рис. 1. Здесь видно, что в одном из текстов нашего корпуса, в котором дважды встречается фамилия “Волобуев”, нейронное распознавание именованных сущностей выделяет её как именованную сущность (тэг I-PER) только в первом случае, но не во втором, где она помечена тэгом O.*

Таким образом, в наш инвертированный индекс попало много “лишних” именованных сущностей, например, слово “москва”. Многие из таких сущностей достаточно частотны и набрали достаточное число вхождений, чтобы ошибочно попасть в список неологизмов. Альтернативы библиотеке NER, построенные на правилах, а не на нейросетях, не показывают значительно лучшего результата в распознавании именованных сущностей.

Поэтому слова пришлось дополнительно проверять на то, являются ли они именованными сущностями. Сначала исключались слова, содержащие в себе не-кириллические символы. Затем для каждого слова проверялось, существует ли посвящённая ему страница в Википедии (для скачивания и парсинга html-страниц использовались те же библиотеки http, urllib и BeautifulSoup, которые использовались для сбора данных). Если да, то слово исключалось. Эта дополнительная проверка так же исключала слова, не являющиеся неологизмами, но попавшие в список из-за низкой частотности (предполагается, что неологизмы ещё не успели «обзавестись» страницей в Википедии, в отличие от не новых, но низкочастотных слов). Следует заметить, что такая проверка хорошо работает только с существительными, так как для глаголов и прилагательных, как правило, не существует страницы в Википедии. После всех мер, направленных на то, чтобы исключить именованные сущности и не-неологизмы, длина списка слов с короткой историей составила 35970 слов (до «ручного» удаления именованных сущностей она составляла 48811 слово). Список слов с короткой историей (включая дату первого и последнего появления для каждого слова) был сохранён в формате csv с разделителем “;”.

Несмотря на все вышеуказанные меры, в результатах однако же оказалось несколько не-неологизмов (“поджарый”, “укладывание” – в основном, это были не-существительные, потому что существительные исключались проверкой на существование страницы в Википедии), именованных сущностей (“холмквист”, “яннопулос”), слов с опечатками (“афганстан”, “ярковыразить”) и несуществующих слов, полученных в результате ошибок при лемматизации (“уотергеть”). Тем не менее, большая часть найденных слов являются неологизмами.

Для каждого из найденных слов был построен график его вхождений в корпус. При построении графиков использовалась библиотека matplotlib. Для построения графиков использовался тот же метод “сглаживания”, что и для автоматического выявления слов с короткой историей. Так как график, учитывающий каждое вхождение слова в корпус, выглядел бы слишком “рваным” и малоинформативным, учитывалось количество вхождений за каждый месяц. Благодаря этому получилось построить более гладкие и информативные графики[[2]](#footnote-2). Пример – график для слова “уотергейт”.



*Рис. 2. Пример графика вхождений слова в корпус – график для слова «уотергейт». По оси x – даты, по оси y – число вхождений за тот или иной месяц.*

1. **Результаты.**

В ходе работы собран корпус новостных газетных текстов за период с 2000 года по 2018 год объёмом 1257850 текстов. Источниками для корпуса послужили четыре нетематических новостных интернет-сайта -- Lenta.ru (734019 текстов), Poilt.ru (323919 текстов), Znak.com (42225 текстов), и сайт информационного агентства Интерфакс interfax.ru (157687 текстов). Для слов корпуса был составлен инвертированный индекс по датам. Объём инвертированного индекса составил 481450 лемм слов.

По инвертированному индексу автоматически были найдены 35970 слов с короткой историей, из которых значительное количество действительно являются неологизмами, а не попали в список по ошибке. Беглый просмотр неологизмов показывает, что значительная часть из них относится к сфере политики (“уотергейт”, “антиглобализм”, “путинщина”), новых технологий (“мэйнфрейм”, “видеосвидетельство”), либо же связана с какими-то ещё новыми реалиями (“наркоторговка”, “информагентство”, “лигалайз”, “суперфорвард”, “милиционный”, “спецтариф”). Список слов (включая дату первого и последнего вхождения для каждого слова) был сохранён.

Для каждого из неологизмов составлен график его вхождений в корпус газетных текстов. Для сглаживания при построении графика учитывалось число вхождений не за каждый день, а за каждый месяц.

1. **Заключение.**

Итак, был проведён эксперимент по автоматическому выявлению неологизмов с короткой историей на материале газетных текстов (источники – сайты четырёх достаточно больших нетематических российских новостных интернет-изданий) 2000-2018 годов. Задача состояла из трёх этапов: сбор данных, составление инвертированного индекса для слов корпуса и автоматическое определение слов с короткой историей по инвертированному индексу (включая построение графиков вхождений слова в корпус для каждого слова с короткой историей). Полученный список содержит в себе 35970 слов.

Тем не менее, в процессе работы мы столкнулись с некоторыми проблемами. Это, во-первых, невозможность полностью отфильтровать именованные сущности (лучшая из существующих библиотек для распознавания именованных сущностей для русского, тем не менее, очень часто даёт ложноотрицательный результат, и даже поиск по Википедии не отфильтровывает все именованные сущности, часто пропуская, например, названия брендов), чтобы они не попали в список неологизмов. Во-вторых, это невозможность полностью отфильтровать слова с опечатками, чтобы они не попали в список неологизмов. Встречаются также несуществующие слова, попавшие в список из-за неправильного результата работы лемматизатора, но их количество незначительно, и им можно пренебречь. Но, несмотря на эти проблемы, используемый метод даёт достаточно хорошие результаты.

Данный метод может в дальнейшем использоваться для обнаружения слов с короткой историей в других корпусах текстов. Собранный корпус газетных текстов может в дальнейшем использоваться для других задач, как и инвертированный индекс по корпусу.

1. **Литература.**

Можарова, Лукашевич 2016 – Можарова В. А., Лукашевич Н. В. Двухэтапный подход к извлечению именованных сущностей // *Труды конференции по искусственному интеллекту КИИ-2016, т.2.*, 2016. - С.81-88.

Ляшевская, Шаров 2009 – Электронная версия издания: О. Н. Ляшевская, С. А. Шаров, Частотный словарь современного русского языка (на материалах Национального корпуса русского языка). М.: Азбуковник, 2009. <http://dict.ruslang.ru/freq.php>

Anh et al. 2017 – Anh L. T., Arkhipov M. Y., Burtsev M. S. Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition. <https://arxiv.org/pdf/1709.09686.pdf>

Gareev et al. 2013 – Rinat Gareev, Maksim Tkachenko, Valery Solovyev, Andrey Simanovsky, Vladimir Ivanov: Introducing Baselines for Russian Named Entity Recognition. *Computational Linguistics and Intelligent Text Processing*. 2013. P. 329-342

Korobov, 2015 – Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages. *Analysis of Images, Social Networks and Texts.* P. 320-332 (2015).

Starostin et al. 2016 – Starostin A. S., Bocharov V. V., Alexeeva S. V., Bodrova A. A., Chuchunkov A. S., Dzhumaev S. S., Efimenko I. V., Granovsky D. V., Khoroshevsky V. F., Krylova I. V., Nikolaeva M. A., Smurov I. M., Toldova S. Y.: FactRuEval 2016: Evaluation of Named Entity Recognition and Fact Extraction Systems for Russian. *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialogue 2016”*

1. **Приложение.**

Код программы расположен по адресу: <https://github.com/vyhuholl/course_work_3rd_year> .

Корпус газетных текстов (в формате pkl) расположен по адресу: <https://drive.google.com/open?id=1I-R4mPcYgjYFkyzNyWazOoikBtSSOxPV> .

Инвертированный индекс корпуса (в формате pkl) расположен по адресу: <https://drive.google.com/open?id=1ffqsJfb_UE0s5qGwtQxKj1Ni5rMRF6jg>

Полученный автоматически список неологизмов расположен по адресу: <https://github.com/vyhuholl/course_work_3rd_year/blob/master/results.csv> .

Графики вхождений каждого из неологизмов в корпус расположены по адресу: <https://drive.google.com/open?id=1FAF-BNCHkLaKK3qOH22GEjcMR0g-n8-y>

1. Метрика оценки качества предсказаний модели, считающаяся по формуле 2 \* (*precision* \* *recall*) / (*precision* + *recall*) , где *precision* – процент элементов, корректно определённых моделью, как правильные, среди всех элементов, определённых моделью, как правильные, а *recall* – процент элементов, определённых моделью, как правильные, среди всех элементов, которые должны быть определены как правильные. Чем ближе F1-мера к единице, тем лучше. [↑](#footnote-ref-1)
2. В библиотеке scipy существует метод spline, предназначенный для сглаживания графиков, но он работает только тогда, когда по оси X числовые данные, а не даты в формате datetime.date, как в нашем случае. [↑](#footnote-ref-2)