Результаты исследования открытых обучаемых систем морфологического и синтаксического анализа на public-тесте GramEval2020

Для исследования были выбраны:

- UDPipe;
- Turku-neural-parser-pipeline (TurkuNLP);
- SyntaxNet;
- MaltParser;
- модуль syntactic parsing из DeepPavlov AI library.

Первые три системы собирались на основе исходных кодов, извлечённых из соответствующих git-репозиториев[1] в период с 06.02.2020 по 16.02.2020. MaltParser v.1.9.2 был загружен в виде готового jar. deeppavlov разворачивался как python-библиотека (19.02.2020).

Для обучения моделей использовались *только* данные из git-репозитория GramEval2020. А именно, файлы:

Train set	Dev set (validation set)
GramEval2020-GSD-train.conllu	GramEval2020-GSD-wiki-dev.conllu
GramEval2020-SynTagRus-train.conllu	GramEval2020-RuEval2017-Lenta-news-dev.conllu
GramEval2020-Taiga-news-train.conllu	GramEval2020-RuEval2017-social-dev.conllu
GramEval2020-Taiga-poetry-train.conllu	GramEval2020-SynTagRus-dev.conllu
GramEval2020-Taiga-social-train.conllu	GramEval2020-Taiga-poetry-dev.conllu
MorphoRuEval2017-JZ-gold.conllu	
MorphoRuEval2017-Lenta-train.conllu	
MorphoRuEval2017-VK-gold.conllu	

Таким образом, из обучающих данных были исключены тексты 17-century.
Кроме того:

- из GramEval2020-Taiga-social-train.conllu было удалено предложение, состоящее из одного слова
  #междуреченский#тобольск#нефтеюганск#тюмень#девушкитюмени#тмн#селфитюмень#новостнаятюмень#тюменьинстаграммная#меняюсебя#урай#ишим#ирбит#ижевск#томск#омск#тавда#нижневартовск#когалым#югорск#нягань#лангепас#сахалин#салехард#инстакрасотки#девушкитюмени#мамавдекрете#декретныебудни#мамыонитакие#горемать
  (оно вызвало затруднения у UDPipe, имеющего ограничение на длину токена);
- во всех файлах в поле XPOS принудительно записан символ подчёркивания.

Train set был объединён в один файл размером ≈ 88 Мб. Dev set — в файл размером ≈ 0.3 Мб.

1 SyntaxNet собран на основе кодов из репозитория https://git.nk2.eu/nk2/syntaxnet_rus.git

UDPipe

При работе с UDPipe почти сразу удалось построить модель с высокими показателями в морфологии. Поэтому бо̀льшая часть экспериментов свелась к попыткам обучения синтаксического анализатора.
UDPipe способен копировать части модели из одного файла в другой. В данном исследовании за основу взята модель russian-syntagrus-ud-2.5-191206.udpipe, которая в дальнейшем модифицировалась, замещением сначала теггера, а затем и парсера.
Поиск «достаточно хорошего» теггера ограничился тремя экспериментами:

- обучение с параметрами по умолчанию (одномодельный tagger);
- обучение с параметрами, использовавшимися авторами UDPipe при обучении модели russian-syntagrus-ud-2.5-191206.udpipe (см. файл params_tagger в архиве udpipe-ud-2.5-191206-reproducible_training.zip);
- обучение двумодельного tagger'a с параметрами по умолчанию.

В исследовании данные модели получили имена m5, m6 и m7, соответственно. Точные параметры командной строки сведены в таблицу ниже.

m5	`./udpipe --train m5.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --parser=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe`
m6	`./udpipe --train m6.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger='models=2;templates_1=tagger;guesser_suffix_rules_1=10;guesser_enrich_dictionary_1=6;guesser_prefixes_max_1=0;use_lemma_1=1;use_xpostag_1=1;use_feats_1=1;provide_lemma_1=0;provide_xpostag_1=1;provide_feats_1=1;prune_features_1=1;templates_2=lemmatizer;guesser_suffix_rules_2=8;guesser_enrich_dictionary_2=5;guesser_prefixes_max_2=4;use_lemma_2=1;use_xpostag_2=0;use_feats_2=0;provide_lemma_2=1;provide_xpostag_2=0;provide_feats_2=0;prune_features_2=1' --parser=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe`
m7	`./udpipe --train m7.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --parser=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger='models=2;templates_1=tagger;use_lemma_1=1;use_xpostag_1=0;use_feats_1=1;provide_lemma_1=0;provide_xpostag_1=0;provide_feats_1=1;templates_2=lemmatizer;use_lemma_2=1;use_xpostag_2=0;use_feats_2=1;provide_lemma_2=1;provide_xpostag_2=0;provide_feats_2=0'`

Испытание этих моделей на тестовом множестве показало, что m7 немного лучше других.

	POS quality			Morpho features			Lemmatization		
	m5	m6	m7	m5	m6	m7	m5	m6	m7
GramEval2020-17cent-dev.conllu	0.9167	0.9207	**0.9246**	0.8187	**0.8277**	0.8244	0.5074	0.5044	**0.5094**
GramEval2020-test-fiction.v02.conllu	0.9671	**0.9691**	0.9652	0.9584	**0.9663**	0.9605	0.9304	0.9383	**0.9453**
ru_taiga-ud-train-poetry.conllu	0.9794	0.9884	**0.9915**	0.9784	0.9907	**0.9923**	0.9694	0.9926	**0.9997**
ru_taiga-ud-dev-poetry.conllu	0.9600	0.9791	**0.9814**	0.9683	0.9922	**0.9940**	0.9323	0.9938	**0.9990**
ru_taiga-ud-test-poetry.conllu	**0.9907**	0.9787	0.9898	0.9676	0.9656	**0.9757**	0.9944	0.9944	**0.9990**
ru_taiga-ud-train-news.conllu	**1.0**	**1.0**	**1.0**	**1.0**	0.9950	**1.0**	**1.0**	**1.0**	**1.0**
ru_taiga-ud-test-news.conllu	**1.0**	0.9937	0.9937	**1.0**	**1.0**	**1.0**	0.9937	**1.0**	**1.0**
ru_taiga-ud-train-social.conllu	**0.9867**	0.9829	0.9854	**0.9743**	0.9721	0.9741	0.9916	0.9943	**0.9978**
ru_taiga-ud-dev-social.conllu	**0.9886**	0.9862	0.9878	**0.9962**	0.9938	0.9950	0.9941	0.9977	**0.9996**
ru_taiga-ud-test-social.conllu	**0.9908**	0.9882	0.9904	**0.9737**	0.9713	**0.9737**	0.9929	0.9938	**0.9981**

Дальнейшие исследования преимущественно опирались на теггер модели m7.

Обучение синтаксического анализатора UDPipe с параметрами по умолчанию для двух вариантов transition system дало отправную точку.

m17	`./udpipe --train m17.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=projective'`
m12	`./udpipe --train m12.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=swap'`

	UAS	
	m17 (proj)	m12 (swap)
GramEval2020-17cent-dev.conllu	0.5540	**0.5708**
GramEval2020-test-fiction.v02.conllu	0.7157	**0.7176**
ru_taiga-ud-train-poetry.conllu	0.6622	**0.6679**
ru_taiga-ud-dev-poetry.conllu	0.6091	**0.6410**
ru_taiga-ud-test-poetry.conllu	**0.7257**	0.7045
ru_taiga-ud-train-news.conllu	**0.8088**	0.7941
ru_taiga-ud-test-news.conllu	0.7861	**0.8427**
ru_taiga-ud-train-social.conllu	0.7290	**0.7390**
ru_taiga-ud-dev-social.conllu	0.6816	**0.7062**
ru_taiga-ud-test-social.conllu	0.7123	**0.7259**

Эксперименты с применением альтернативных векторных представлений позволяют немного улучшить качество анализа. Были опробованы векторная модель из Turku models_ru_syntagrus (от 05.09.2018) и модель, построенная на данных корпуса PaRuS (word2vec с параметрами size 100, window 1, min-count 50).

m8	`./udpipe --train m8.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=projective;embedding_form=100;embedding_form_file=turku.vectors'`
m9	`./udpipe --train m9.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=swap;embedding_form=100;embedding_form_file=turku.vectors'`
m18	`./udpipe --train m18.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=projective;embedding_form=100;embedding_form_file=parus.vectors'`
m19	`./udpipe --train m19.udpipe trainset.conllu --heldout=devset.conllu --tokenizer=from_model=file:russian-syntagrus-ud-2.5-191206.udpipe --tagger=from_model=file:m7.udpipe --parser='transition_system=swap;embedding_form=100;embedding_form_file=parus.vectors'`

	UAS					
	m17 (proj)	m12 (swap)	m8 (proj, turku)	m9 (swap, turku)	m18 (proj, parus)	m19 (swap, parus)
GramEval2020-17cent-dev.conllu	0.5540	0.5708	**0.5738**	0.5470	0.5599	0.5470
GramEval2020-test-fiction.v02.conllu	0.7157	0.7176	0.7067	**0.7335**	0.6998	0.7147
ru_taiga-ud-train-poetry.conllu	0.6622	0.6679	0.6703	**0.7029**	0.6768	0.6952
ru_taiga-ud-dev-poetry.conllu	0.6091	0.6410	0.6323	**0.6697**	0.6295	0.6664
ru_taiga-ud-test-poetry.conllu	0.7257	0.7045	0.7063	0.7257	0.7239	**0.7460**
ru_taiga-ud-train-news.conllu	0.8088	0.7941	0.8382	0.8382	0.8382	**0.8750**
ru_taiga-ud-test-news.conllu	0.7861	**0.8427**	0.7924	0.8176	0.8113	0.8238
ru_taiga-ud-train-social.conllu	0.7290	0.7390	0.7318	**0.7768**	0.7327	0.7727
ru_taiga-ud-dev-social.conllu	0.6816	0.7062	0.7051	**0.7478**	0.6900	0.7310
ru_taiga-ud-test-social.conllu	0.7123	0.7259	0.7266	**0.7568**	0.7133	0.7410

В большинстве случаев swap оказывается лучше projective.

MaltParser

MaltParser решает только задачу синтаксического анализа, поэтому в исследовании он использовался в конвейере с UDPipe. Теггер использовался из модели m7 (см. выше про модели для UDPipe).
Команда конвейера в общем виде:

```
./udpipe --input=conllu --output=conllu --tag m7.udpipe <INPUT.conllu>  | \
java -Xms2g -Xmx4g -jar maltparser-1.9.2.jar -c <MODEL> -o <OUTPUT.conllu> -m parse
```

MaltParser не использует validation set, поэтому для его обучения был сформирован файл data.conllu, полученный путём объединения train set и dev set.
Для MaltParser было построено 4 модели со следующими параметрами:

m13	`java -Xmx8000m -jar maltparser-1.9.2.jar -c ge_ru_nivreeager -i data.conllu -m learn -l liblinear -a nivreeager`
m14	`java -Xmx32g -jar maltparser-1.9.2.jar -c ge_ru_covnonproj -i data.conllu -m learn -l liblinear -a covnonproj`
m15	`java -Xmx8000m -jar maltparser-1.9.2.jar -c ge_ru_stackeager -i data.conllu -m learn -l liblinear -a stackeager`
m16	`java -Xmx8000m -jar maltparser-1.9.2.jar -c ge_ru_stacklazy -i data.conllu -m learn -l liblinear -a stacklazy`

Результаты уступают UDPipe, хотя у MaltParser ещё остаются возможности для улучшения UAS за счёт пересмотра системы признаков (мы использовали признаки по умолчанию).

	UAS			
	m13	m14	m15	m16
GramEval2020-17cent-dev.conllu	0.3072	0.3102	0.3458	**0.3548**
GramEval2020-test-fiction.v02.conllu	**0.4691**	0.4433	0.4453	0.4493
ru_taiga-ud-train-poetry.conllu	**0.4522**	0.4348	0.4318	0.4288
ru_taiga-ud-dev-poetry.conllu	**0.3920**	0.3861	0.3744	0.3776
ru_taiga-ud-test-poetry.conllu	**0.6805**	0.6325	0.6463	0.6574
ru_taiga-ud-train-news.conllu	**0.7867**	0.7279	0.7720	0.7720
ru_taiga-ud-test-news.conllu	**0.7295**	0.7106	0.6540	0.6352
ru_taiga-ud-train-social.conllu	**0.6195**	0.5880	0.5899	0.5979
ru_taiga-ud-dev-social.conllu	0.6057	0.5898	**0.6090**	**0.6090**
ru_taiga-ud-test-social.conllu	**0.6137**	0.5779	0.5775	0.5767

SyntaxNet

У SyntaxNet много возможностей для настройки.
В данном исследовании использовалась конфигурация, созданная, вообще говоря, для английского. Но беглый взгляд по системе признаков навёл на мысль, что обучение не безнадёжно. Хотя парсер и не будет ничего знать о морфологических атрибутах, он сможет строить свои гипотезы на основе лексики, информации о частях речи и уже принятых синтаксических решениях. Что делает его сопоставимым с MaltParser.
Параметры обучения были заимствованы в том же репозитории. В результате была обучена модель решающая задачи предсказания части речи и синтаксического разбора. На её основе было сделано два сабмита:

- m10 — оценка построенной модели;
- m26 — теггирование по m7 + синтаксис по m10.

Результаты представлены в следующей таблице.

	POS quality		UAS	
	m10	m26=m7	m10	m26
GramEval2020-17cent-dev.conllu	0.8543		0.4400	**0.5064**
GramEval2020-test-fiction.v02.conllu	0.9254		0.6361	**0.6819**
ru_taiga-ud-train-poetry.conllu	0.9442		0.6330	**0.6581**
ru_taiga-ud-dev-poetry.conllu	0.9340		0.5863	**0.5929**
ru_taiga-ud-test-poetry.conllu	0.9418		0.7128	**0.7331**
ru_taiga-ud-train-news.conllu	0.9779		0.8161	**0.8382**
ru_taiga-ud-test-news.conllu	0.9937		**0.8050**	**0.8050**
ru_taiga-ud-train-social.conllu	0.9488		0.7191	**0.7327**
ru_taiga-ud-dev-social.conllu	0.8908		0.6746	**0.6981**
ru_taiga-ud-test-social.conllu	0.9509		0.7201	**0.7307**

Перспективным видится эксперимент обучения SyntaxNet с полноценным теггером. Идеи системы признаков можно найти онлайн (см. brain_morpher_features). Также имеет смысл попробовать обучать один только парсер (без теггера частей речи), т. к. при совместном обучении парсер учится на данных, скорректированных обученным теггером частей речи.

TurkuNLP

Полноценно исследовать этот инструмент не удалось ввиду отсутствия под рукой GPU (lemmatizer без GPU обучаться отказался). Тем не менее получилось обучить теггер и синтаксический анализатор. Исследовались две программные конфигурации:

- m11 — теггер и парсер обучены (векторные представления заимствованы из models_ru_syntagrus от 05.09.2018) + лемматайзер взят из той же models_ru_syntagrus;
- m21 — теггер и лемматайзер из m7 + синтаксис из m11.

	POS quality		Morpho features		Lemmatization		UAS	
	m11	m21	m11	m21	m11	m21	m11	m21
GramEval2020-17cent-dev.conllu	0.8860	**0.9246**	**0.8422**	0.8244	**0.5133**	0.5094	0.6412	**0.6858**
GramEval2020-test-fiction.v02.conllu	0.9552	**0.9652**	0.9596	**0.9605**	0.9314	**0.9453**	**0.8210**	0.8190
ru_taiga-ud-train-poetry.conllu	0.9521	**0.9915**	0.9373	**0.9923**	0.9061	**0.9997**	0.7800	**0.8036**
ru_taiga-ud-dev-poetry.conllu	0.9443	**0.9814**	0.9440	**0.9940**	0.9032	**0.9990**	0.7521	**0.7692**
ru_taiga-ud-test-poetry.conllu	0.9529	**0.9898**	0.9244	**0.9757**	0.9353	**0.9990**	0.7940	**0.8153**
ru_taiga-ud-train-news.conllu	0.9779	**1.0**	0.9681	**1.0**	0.9779	**1.0**	0.9044	**0.9338**
ru_taiga-ud-test-news.conllu	**0.9937**	**0.9937**	0.9810	**1.0**	0.9685	**1.0**	0.8679	**0.8867**
ru_taiga-ud-train-social.conllu	0.9578	**0.9854**	0.9507	**0.9741**	0.9497	**0.9978**	0.8116	**0.8316**
ru_taiga-ud-dev-social.conllu	0.9269	**0.9878**	0.9588	**0.9950**	0.8918	**0.9996**	0.8318	**0.8604**
ru_taiga-ud-test-social.conllu	0.9636	**0.9904**	0.9462	**0.9737**	0.9464	**0.9981**	0.8047	**0.8319**

DeepPavlov AI

Исследовался только синтаксический модуль, обученный на train/dev-сетах GramEval2020. В качестве сабмита использовалась гибридная модель: теггер и лемматайзер из m7 + синтаксис из deeppavlov.

	UAS
	m31 (deeppavlov)
GramEval2020-17cent-dev.conllu	0.7561
GramEval2020-test-fiction.v02.conllu	0.8359
ru_taiga-ud-train-poetry.conllu	0.8239
ru_taiga-ud-dev-poetry.conllu	0.8084
ru_taiga-ud-test-poetry.conllu	0.8799
ru_taiga-ud-train-news.conllu	0.9044
ru_taiga-ud-test-news.conllu	0.8867
ru_taiga-ud-train-social.conllu	0.8489
ru_taiga-ud-dev-social.conllu	0.8832
ru_taiga-ud-test-social.conllu	0.8389

Выводы

Из исследованных конфигураций наилучший результат (0.9353) по показателю Overall quality получила модель m31 (гибрид UDPipe и DeepPavlov). К сожалению в public-тесте отсутствует LAS-показатель, поэтому о возможностях синтаксических анализаторов данное исследование говорит не так уж много.

Исследование показывает, что в части морфологического анализа UDPipe представляет собой добротное решение (зазор между UDPipe и 1.0 не так уж велик). В синтаксисе есть над чем поработать.

20.02.2020
Игорь Трофимов
ИЦИИ ИПС РАН