Projekt wynikanie – temat 5

Wojciech Bartoszek, Albert Dziugieł, Maciej Gorczyca, Bogdan Jastrzębski

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska

Etap wstępny

0.1 Zadanie

DANE: czasownik (cechy + embedding)
WYJŚCIE: sygnatura czasownika (+,-,o / +,-.o)

0.2 Technologia

Projekt zostanie zrealizowany z użyciem języka Python. Dodatkowo wykorzystana zostanie biblioteka PyTorch jako narzędzie usprawniające prace z technikami uczenia maszynowego.

0.3 Podział prac

Tabela 1 zawiera podział prac nad poszczególnymi częściami projektu.

Tabela 1. Podział prac nad projektem.

Osoba	Zadania
Wojciech Bartoszek	Przygotowanie sprawozdania
Albert Dziugieł	Implementacja
Maciej Gorczyca	Ocena wyników
Bogdan Jastrzębski	Analiza problemu

0.4 Dotychczasowe postępy innych studentów

Na dysku Google zostały udostępnione dwa repozytoria studentów z poprzednich lat:

- `implication-annotator`
- `wynikanie`

Pierwszy z nich jest realizacją projektu numer jeden – dotyczącego podziału zdania na części oraz cechy zdania takie jak:

1. Podmiot
2. Czasownik
3. Complementaizer
4. Wartości morfologiczne
5. Typ zdania
6. Negacja

Drugi z repozytoriów zawiera implementację drugiego projektu, który rozpoznaje klasę typu wynikania E, C, N lub ?.

W temacie numer pięć, możemy wykorzystać część implementacji z repozytorium "wynikanie" zawierającą wczytywanie embeddingów oraz cech czasowników.

1 Analiza problemu

Naszym zbiorem danych jest zestaw zdań podrzędnie złożonych z "że". Pierwsza część zdania zawiera czasownik, druga część zawiera pewne sformułowanie, zdarzenie. Głównym zadaniem projektu jest stworzenie modelu, który na podstawie tego zdania oraz informacji o czasowniku z części nadrzędnej określi czy część podrzędna jest prawdziwa, fałszywa, bądź nie można tego stwierdzić.

Jako przykład może posłużyć zdanie "*Antoni widział, że Aneta jadła kolację.*". Na podstawie czasownika *widzieć* w czasie przeszłym można stwierdzić, że zdanie podrzędne jest prawdziwe i podana sytuacja się wydarzyła. Cecha czasownika, która określa czy zdanie podrzędne jest prawdziwe nosi nazwę – **werydyczność**.

Projekt został podzielony na kilka mniejszych podprojektów. Zadaniem naszego zespołu jest stworzenie modelu, który jako wejście przyjmuje czasownik oraz jego cechy. Stworzony model ma predykować werydyczność wejściowego czasownika.

2 Cel prac projektowych

Celem projektu jest stworzenie systemu, bazującego na sztucznej inteligencji, który będzie rozpoznawał prawdziwość części podrzędnych zdań złożonych. Jest to jedno z wielu zadań, z gałęzi nauki przetwarzania języka naturalnego (NLP).

Analiza języka naturalnego jest bardzo szeroką dziedziną, która jest bardzo przydatna w dziedzinie sztucznej inteligencji. Stworzenie systemu, który rozpoznaje język naturalny znacząco ułatwia wykonywanie wielu zadań oraz wesprzeć rozwój badań, przykładami mogą być: robotyka, przetwarzanie dużych ilości danych, obsługa urządzeń medycznych, internet rzeczy i wiele innych.

3 Przegląd dotychczasowych rozwiązań SOTA (*state-of-the-art*)

Problemem werydyczności czasownika jest przedmiotem badań wielu opracowań zarówno w dziedzinie lingwistyki, jak i przetwarzania języka naturalnego (NLP). Sam problem należy do dziedziny problemów NLI (*Natural language inference*) zajmującej się wnioskowaniem o wystąpieniu pewnego faktu (hipotezy) na podstawie podanego zdania (przesłanki) [1].

W dużej ilości prac werydyczność czasownika rozpatrywana jest jako zjawisko lingwistyczne. Stąd, zakłada się w nich, że rodzaj werydyczności danego czasownika determinowany jest przez inne jego cechy – sygnaturę. W takim przypadku, wcześniej przygotowana wiedza o sygnaturze czasownika wykorzystywana jest do stworzenia systemu wnioskowania semantycznego. Stworzony model wykorzystuje dane o sygnaturze jako znane *a priori* metadane.

Innym podejściem jest podejście *pragmatyczne*, które zakłada, że niektóre czasowniki wskazują bezpośrednio na wystąpienie danego faktu, niezależnie od ich znaczenia leksykalnie-semantycznego. Oznacza to, że w zależności od kontekstu wypowiedzi mówcy, dane zdanie z czasownikiem, który zazwyczaj nie jest uznawany za odpowiedni do wnioskowania, może pozwolić jednoznacznie wywnioskować o tym, czy dana hipoteza jest spełniona. Na przykład zdanie "*Nie odmówił składania zeznań*", może pozwolić na sklasyfikowanie hipotezy "*Złożył zeznania*" jako prawdziwej, o ile nie pojawiły się dodatkowe stwierdzenia dot. rozpatrywanej hipotezy – np. "*...po prostu nie miał nic do powiedzenia*" Niestety takie podejście jest zbyt mało systematyczne i subiektywnie zależne od osoby anotującej dane, aby mogło być wykorzystane do automatycznego wnioskowania [2].

Próba rozwiązania podobnego problemu została przeprowadzona w publikacji [3]. W tym przypadku wykorzystany został zbiór Fact Bank, który został ponownie anotowany przez osoby zatrudnione do tego celu przez twórców artykułu. Celem wykonania automatycznej klasyfikacji werydyczności czasowników twórcy wykorzytstali Maximum Entropy Classifier. Dodatkowym problemem były różne klasyt werydyczności przypisane przez kolejnych anotatorów. Problem ten został rozwiązany przez potraktowanie wszystkich anotacji jako różnych elementów zbioru traningowego. Średnia otrzymana wartość miar mikro-średniej dla wszystkich klas wyniosła 78.6% dla zbioru treningowego i 83.0% dla zbioru testowego.

4 Opis zbioru danych

Otrzymany zbiór danych zawierał 2596 anotowanych zdań z czasownikami, z czego różnych czasowników było 368. Średnio zbiór danych zawierał 5-6 rekordów dla pojedynczego czasownika. Dodatkowo, niektóre czasowniki podane były w formie sekwencji różnej długości: np. "*być pewnym, że*", "*dać do zrozumienia, że*". Do wykonania zadania klasyfikacji wykorzystane zostały następujące kolumny ze zbioru:

- `verb` – czasowniki zostały zanurzone w wektor liczb (*embedding*), w przypadku wyrażeń zsumowano embeddingi wszystkich słów danego wyrażenia,
- `GOLD <T, H>` – *General Ontology for Linguistic Description*, kodowanie lingwistyczne, możliwe wartości: **C**, **E**, **N**.
- `GOLD <T1, H>` – kodowanie lingiwstyczne dla zanegowanego zdania T, (1823 rekordy oznaczone jako *nie dotyczy*).
- `verb - main semantic class` - klasa semantyczna (wszystkie rekordy mają wartości,
- `verb - second semantic class` - dookreślenie klasy semantycznej czasownika (1993 rekordów z brakami),
- `verb - third semantic class` - dookreślenie klasy semantycznej (2489 rekordów z brakami).

Wartości były klasyfikowane na klasy z dwóch kolumn określających pozytywną i negatywną werydyczność, z licznością przedstawicieli poszczególnych klas:

- `verb - veridical (positive environment)`

Tabela 2. Liczności elementów klasy werydyczności pozytywnej.

Klasa	Liczność
o	1738
+	796
-	33
+?	14
o?	13
?	2

- `verb - veridical (negatice environment)`

Tabela 3. Liczności elementów klasy werydyczności negatywnej.

Klasa	Liczność
o	1872
+	589
?	97
-	22
1?	9
+?	7

W dalszych rozważaniach połączone zostały kalasy "+" z klasą "+?", "o" z "o?" oraz "1?" z "?". Oprócz tego występuje spora rozbieżność pomiędzy licznościami klas w obu przypadkach. Obiektów klasy o jest zdecydowanie więcej niż obiektów pozostały klas, aby poradzić sobie z tym problemem przy ocenie jakości klasyfikacji została wykorzystana miara F1.

5 Opis rozwiązania

Główną trudnością w tym projekcie było z pewnością to, że nasze dane są bardzo szerokie. Embedding ma 300 wymiarów, razem z pozostałymi nasz zbiór danych ma kilkanaście więcej. Jednocześnie po selekcji wymiarów i ograniczeniu się do unikalnych wierszy zostało mniej niż 600, dlatego pula dostępnych rozwiązań jest bardzo mała ze względu na duże prawdopodobieństwo przeuczenia.

W trakcie prac zostały przetestowane następujące podejścia:

- trenowanie modelu na danych bez embeddingów
- trenowanie modelu na danych w postaci samych embeddingów
- trenowanie modelu na danych w embeddingów z dodatkowymi informacjami

Zaletą pierwszego podejścia jest to, że nie byliśmy ograniczeni co do wyboru rozwiązania. Jednocześnie pozbawiliśmy zbiór potencjalnie bardzo dużej puli informacji. Ostatecznie okazało się, że klasyfikator oparty tylko na podstawowych informacjach miał bardzo słabe wyniki, podejście to zostało we wczesnym etapie projektu zaniechane.

Same embeddingi z pewnością dostarczyły nowych informacji do zbioru. Jakość predykcji nie urosła bardzo przy modelu opartym na nich samych, natomiast model łączony (podejście 3) był już zdecydowanie lepszy.

Do zanurzenia czasowników ze zbioru danych wykorzystano embeddingi typu **fastText** dla języka polskiego [4]. W rozważanym problemie istotną trudnością były wyrażenia czasownikowe składające się ze zmiennnej liczby słów. Zmienna liczba słów powodowała, że embeddingi również nie miały stałej długości, zatem dane były sekwencyjne. Klasyczne algorytmu z dziedziny uczenia maszynowego nie są przystosowane do wektorów wejściowych o zmiennej długości, więc problem ten trzeba było rozwiązać.

W takim wypadku można np. wykorzystać sieci konwolucyjne (splotowe, convolutional) albo sekwencyjne (GRU, LSTM, RNN), jednak bardzo mały rozmiar zbioru uniemożliwiał skorzystanie z takich technik bez transfer learning. Transfer learning to termin, który oznacza wykorzystanie wytrenowanych wcześniej funkcji/klasyfikatorów (np. sieci neuronowych) w nowym problemie, wykorzystując stare wagi, bądź też na ich podstawie wyznaczając nowe (tunning).

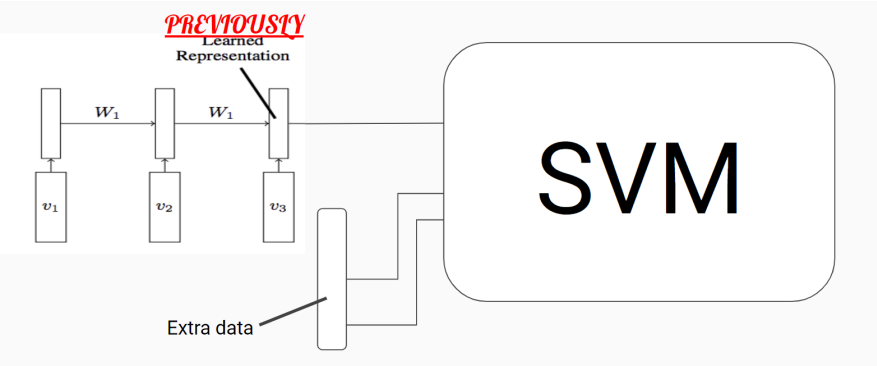
Początkowo skorzystaliśmy z autoencoder'a LSTM. Był to bardzo dobry pomysł w teorii, jednak w rzeczywistości nie dał dobrych rezultatów. Okazało się, że lepszym rozwiązaniem jest wybranie słowa kluczowego, albo zwykłe uśrednienie embeddingów.

Ponieważ rozpatrywane ze zbioru danych cechy czasowników miały charakter wartości pochodzących ze skończonego zbioru, niebędących w żadnej relacji porządku to zostały one zakodowane podejściem one-hot. W projekcie należało dokonać klasyfikacji wieloetykietowej tzn. każdemu czasownikowi trzeba było przyporządkować klasę oznaczającą jego werydyczność w pozytywnym oraz w negatywnym środowisku. Aby móc rozwiązać zadania za pomocą pojedynczego klasyfikatora (mimo, że trzeba przewidzieć dwie klasy dla każdego czasownika), przekształcono etykiety w ten sposób, że zbiór wynikowych etykiet stanowią

wszystkie kombinacje par etykiet z dwóch rozważanych zbiorów klas (werydyczność w pozytywnym oraz negatywnym środowisku). Te obserwacje ze zbioru dla których którakolwiek z klas była '?' zostały pominięte w eksperymencie.

W trakcie prac przetestowane zostały modele:

- MLP - implementacja: pytorch
- LSTM-SVM - model LSTM-SVM to model, który uczył jednocześnie LSTM i SVM z krenelem gausowskim. Bardzo ciekawe, autorskie podejście. Niestety, nie dało zbyt dobrych rezultatów, głównie przez wymiar zbioru danych. Implementacja: pytorch.
- KNeighborsClassifier: SKLEARN
- RandomForestClassifier: SKLEARN
- MLPClassifier: SKLEARN
- AdaBoostClassifier: SKLEARN
- GaussianProcessClassifier: SKLEARN
- GradientBoostingClassifier: SKLEARN
- SVC: SKLEARN



W celu znalezienia modelu o jak najlepszych rezultatach zastosowano podejście wyszukiwania wyczerpującego *(eng. Exhaustive Grid Search)*. Idea tej metody polega na pełnym przeszukaniu pewnego hiperześcianu, którego wymiarami są hiper-parametry klasyfikatorów i zwrócenie tej konfiguracji , która maksymalizuje pewną funkcje celu.

6 Analiza wyników, wnioski

W procesie uczenia zastosowano następujący podział na zbiór trenujący i testowy. Zadbano o to, aby obserwacji ze zbioru danych związane z danymi czasownikiem znajdowały się wszystkie tylko w jednym ze zbiorów. Obecność czasownika zarówno w zbiorze trenującym i testowym jest nieprawidłowa - powoduje nieporządane podobieństwo między obydwoma zbiorami (embeddingi stanowią znaczną część wektora wejściowego).Tabela 4 zawiera rezultaty przeprowadzonych eksperymentów. Eksperymentów dokonano przy podziale zbioru danych na trenujący i testowy w stosunku 4:1. Wartości metryk precision, recall oraz f1 obliczono jako średnią ważoną rezultatów poszczególnych klas.

Tabela 4. Wyniki eksperymentów przeprowadzonych na różnych klasyfikatorach

Klasyfikator	**Parametry**	**Zb. treningowy**	**Zbiór testowy**		
		F1	**Precision**	**Recall**	**F1**
`KNeighborsClassifier`	'n_neighbors': 3	0.720	0.776	0.795	0.782
`RandomForestClassifier`	'criterion': 'gini', 'max_features': 'auto', 'n_estimators': 110	0.701	0.599	0.697	0.632
`MLPClassifier`	'activation': 'relu', 'alpha': 0.001, 'learning_rate': 'invscaling', 'solver': 'adam'	0.823	0.826	0.860	0.839
`AdaBoostClassifier`	'n_estimators': 100	0.503	0.423	0.648	0.512
`GaussianProcessClassifier`		0.746	0.766	0.795	0.780
`GradientBoostingClassifier`	'max_depth': 8, 'n_estimators': 830	0.763	0.747	0.795	0.757
`SVC`	'C': 7.78, 'gamma': 0.00633	0.804	0.811	0.844	0.823

Jak widać, najlepsze wyniki osiągnął MLP, a po nim SVC. Obydwa klasyfikatory osiągnęły wynik F1 ponad 0.8. Najsłabiej podziałał ADA-Boost.

7 Podsumowanie

Główną trudność problemu stanowiła bardzo niewielka ilość obserwacji w stosunku do wymiarowości danych. Oznacza to, że w praktyce nawet regresja liniowa potrafi się przetrenować na takim modelu, nie wspominając o bardziej złożonych klasyfikatorach. Ze wszystkich przetestowanych rozwiązań, które zostały

przetestowane, okazało się, że najlepsze wyniki dały proste klasyfikatory i proste uśrednianie embeddingów. Dobranie odpowiednich parametrów do SVM i MLP pozwoliło na uzyskanie zdecydowanie lepszych wyników niż próba pokonania sekwencyjności danych poprzez zastosowanie bardziej skomplikowanych rozwiązań.

Literatura

1. NLP-progress. Natural language inference.
2. Alexis Ross and Ellie Pavlick. How well do NLI models capture verb veridicality? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2230–2240, Hong Kong, China, November 2019. Association for Computational Linguistics.
3. M.-C Marneffe, Christoper Manning, and Christopher Potts. Veridicality and utterance understanding. pages 430 – 437, 10 2011.
4. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.