Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa magisterska

na kierunku Informatyka
w specjalności Systemy Informacyjno-Decyzyjne

Klasyfikacja tekstów ironicznych

Filip Lewczak
Numer albumu 259024

promotor
dr inż. Mariusz Kamola

WARSZAWA 2020

Klasyfikacja tekstów ironicznych

Streszczenie.

Praca dąży do zaproponowania modelu zdolnego do klasyfikacji tekstów o charakterze ironicznym. Aby to było możliwe, praca w pierwszej kolejności analizuje istniejące już podejścia do rozwiązania takiego zadania klasyfikacji. By następnie w oparciu o zdobytą wiedzę zaproponować kilka architektur sieci najlepiej nadających się do rozwiązania tego problemu.

Praca skupia się na analizie sieci CNN i sieci opartych na warstwach LSTM oraz na doborze konfiguracji hiper parametrów pozwalającej osiągnąć jak najlepszą dokładność klasyfikacji. Aby to było możliwe w pierwszej kolejności dokonywany jest preprocessing danych. W trakcie niego część elementów zbioru jest zastępowana bardziej generalnymi oznaczeniami, aby zminimalizować poziom szumu w zbiorze uczącym. Ponadto w trakcie tego kroku następuje konwersja emotikon i hasztagów do formy lepiej interpretowalnej przez modele językowe. W następnej kolejności praca skupia się na transformacji danych w formie słów do przestrzeni liczbowej, która może być interpretowana przez sieci neuronowe. W tym celu dokonana jest analiza istniejących sposobów uzyskiwania embeddingów i wybór kilku z nich, najlepiej pozwalających na oddanie cech słów w przestrzeni wektorowej.

Praca, oprócz zaproponowania modelu pozwalającego na detekcję ironii, skupia się także na analizie wpływu oznaczeń morfosyntaktycznych na jakość klasyfikacji. Stara się odpowiedzieć na pytanie, czy taka informacja jest istotna dla sieci w procesie nauki, czy wprowadza może tylko nie istotne szumy.

Słowa kluczowe: ironia, NLP, analiza morfosyntaktyczna

Classification of ironic texts.

Abstract. //TODO

Keywords: XXX, XXX, XXX

załącznik nr 3 do zarządzenia
nr 28 /2016 Rektora PW

Politechnika Warszawska

……..
miejscowość i data

……..
imię i nazwisko studenta
……..
numer albumu
…….…….
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

Spis treści

1. Wstęp

1.1. Cel pracy

Celem pracy było stworzenie modelu pozwalającego na detekcję ironii i sarkazmu w tekście. Przy czym praca, skupia się bardziej na analizie krótkich form wypowiedzi, składających się z nie więcej niż kilku zdań. Dłuższe formy wypowiedzi o charakterze ironicznym takie jak na przykład felietony wymagają innego typu analizy i nie są uwzględniane w ramach pracy. Ponadto w ramach badań podjęta jest analiza wpływu cech morfosyntaktycznych na jakość klasyfikacji modelu. Analiza ta wynika z chęci zweryfikowaniem tezy, że dodatkowe informacje na temat roli słowa w zdaniu pozwolą na ujednoznacznienie znaczenia słowa, co przełoży się na lepszą jakość analizy tekstu w tym konkretnym zadaniu klasyfikacji.

Poprzez cechy morfosyntaktyczne rozumiane są między innymi informację o tym[1]:

- Do jakiej części mowy należy słowo
- Czy słowo występuje w liczbie pojedynczej czy mnogiej
- W jakim przypadku występuje słowo
- W jakim czasie występuje słowo

1.2. Ironia

Ironia jest to sposób wypowiadania się, oparty na zamierzonej niezgodności, najczęściej przeciwieństwie, dwóch poziomów wypowiedzi: dosłownego i ukrytego. W potocznych wypowiedziach utożsamia się ją z zawoalowaną kpiną, złośliwością, czy wyśmiewaniem.

Do najpopularniejszych rodzajów ironii należą[2][3]:

- Ironia sytuacyjna - wiąże się z rozbieżnością między tym jakie są oczekiwania co do tego co powinno się wydarzyć, a tym co faktycznie się wydarzyło np. “Na konferencji na temat IT nie było dostępu do Internetu”
- Ironia werbalna - występuje, gdy intencja (przekaz) osoby wypowiadającej się jest inna niż wynikałoby z dosłownego rozumienia słów w wypowiedzi np. “Nie mogę się już doczekać by wreszcie przeczytać ten siedmiuset stronicowy raport”
- Dramatyczna ironia - jest to rodzaj ironii popularny w filmach, książkach i sztukach teatralnych, występuje, gdy widz/czytelnik posiada kluczowa informację, której nie zna bohater utworu np. “tragiczna śmierć Romea z utworu ‘Romeo i Julia’, który nie jest świadomy, że jego ukochana tak naprawdę nie umarła, a zapadła w stan tylko z pozoru przypominający śmierć”

Zjawiskiem często łączącym się z ironią jest sarkazm. Sarkazm jest formą ironii mającej na celu skrytykowanie zaistniałej sytuacji lub obrażenie danej osoby.

- Przykład ironii: “Super, ktoś poplamił moją nową sukienkę”
- Przykład sarkazmu: “Nazywasz to coś dziełem sztuki?”

1.3. Analiza literatury

Zagadnienie klasyfikacji ironii jest dość popularnym problemem wśród badaczy zajmujących się analizą języka naturalnego ze względu na konieczność wychwycenia nie tylko dosłownego znaczenia danej wypowiedzi, ale także detekcję i analizy ewentualnego podtekstu.

Pierwsze prace z tej dziedziny opierały się na analizie sentymentu poszczególnych słów. Opierały się na przeświadczeniu, że jeśli wypowiedź utrzymuje stały charakter emocjonalny w postaci pozytywnego lub negatywnego wydźwięku nie zawiera ono w sobie ironii. Natomiast jeśli pojawiały się fragmenty o przeciwnej polaryzacji, to zdanie z dużym prawdopodobieństwem można było klasyfikować jako ironiczne. Takie metody skupiały się przede wszsytkim na kryteriach ilościowych, nie uwględniając większej istotności słów kluczowych. Z tego powodu metody tego typu były dość zawodne.

Inne podejście, zaprezentowane w pracy [4] opierało się na spostrzeżeniach, że w zdaniach sarkastycznych słowa o pozytywnym wydźwięku kontrastują z sytuacjami nacechowanymi negatywnie. W oparciu o tą tezę badacze zaproponowali algorytm systematycznie uczący się dwóch grup sformułowań:

- słów o pozytywnym wydźwięku (oznaczony jako PS)
- sformułowań świadczących o sytuacjach nacechowanych negatywnie (oznaczony jako NS)

Algorytm rozpoczyna swoje działanie od tylko jednego słowa ‘love’ w zbiorze słów PS. Następnie dokonywane są kolejne kroki:

1. Wyszukiwanie w zbiorze danych rekordów zawierających słowo ze zbioru PS
2. Dla każdego znalezionego rekordu w otoczeniu słowa ze zbioru PS wyznaczany jest 1-gram,2-gram oraz 3-gram.
3. Ze stworzonych n-gramow wybierany jest ten, który występuje najczęściej dla rekordów sarkastycznych w otoczeniu słów ze zbioru PS i jest dodawany do zbioru NS
4. Następnie dokonywane jest wyszukiwanie w zbiorze danych, takich rekordów, które zawierają słowa ze zbioru NS
5. Dla każdego znalezionego rekordu w otoczeniu słowa ze zbioru NS wyznaczany jest 1-gram,2-gram oraz 3-gram.
6. Ze stworzonych n-gramow wybierany jest ten, który występuje najczęściej dla rekordów sarkastycznych w otoczeniu słów ze zbioru NS i jest dodawany do zbioru PS
7. Powrót do kroku 1, aż do momentu, gdy zbiory PS i NS będą wystarczająco liczne.

Dzięki takiemu podejście możliwe jest zidentyfikowanie najczęściej pojawiających się kombinacji wskazujących na występowanie sarkazmu i w oparciu o nie dokonać predykcji czy dana wypowiedź ma charakter sarkastyczny. Istotną wadą takiego podejścia jest

dość wąskie okno w ramach dokonywana jest analiza zdania, a także brak uwzględniania kulturowego znaczenia poszczególnych słów.

Ze względu na obserwowaną lepszą skuteczność klasyfikacji, późniejsze prace skupiały się przede wszystkim na wykorzystaniu sieci neuronowych do zagadnienia rozpoznawania ironii.

Kluczowym elementem w przypadku wykorzystania sieci neuronowych jest preprocessing danych. Dlatego prace [5] [6] [7] z tej dziedziny skupiały się w pierwszej kolejności na usunięciu lub zastąpieniu specjalnymi oznaczeniami takich informacji jak linki do zewnętrznych portali oraz oznaczenie użytkownika. Ponadto tekst często był normalizowany poprzez wykorzystanie lematyzacji i stemmingu co miało na celu zmniejszenie liczby unikalnych słów.

Kolejnym istotnym elementem w ramach przetwarzania tekstu, wykorzystującego sieci neuronowe, był sposób konwersji słów do przestrzeni liczbowej. W publikacjach autorzy wykorzystywali różne metody, między innymi powszechnie znany word2vec, czy też zyskujący coraz większa popularność ELMo. Tak przetworzone dane były wprowadzane do sieci zarówno opartych o warstwy konwolucyjne, jak i warstwy typu LSTM. Jakość klasyfikacji dla różnych architekrur opartych o te warstwy była dość podobna z nieznaczną przewagą dla sieci opartych na warstwach LSTM.

2. Zbiory danych

W ramach pracy zostały wykorzystane dwa zbiory danych. Pierwszy zbiór danych (w pracy oznaczony jako zbiór A) został udostępniony w ramach inicjatywy SemEval. Inicjatywa ta ma na celu rozwój szeroko pojętej analizy i przetwarzania języka naturalnego. Zbiór ten składa się 4618 próbek oznaczonych tagiem klasyfikującym rekord jako ironiczny lub nie. Dane pochodzą z Twittera i reprezentują trzy różne typy ironii:

- Słowną werbalną stworzoną poprzez wykorzystanie przeciwnej biegunowości słów (polarity contrast):
  - Przykład: I love waking up with migraines #not :'(
- Słowną werbalną stworzoną bez wykorzystania przeciwnej biegunowości:
  - Przykład: Human brains disappear every day. Some of them have never even appeared. #brain #humanbrain #Sarcasm
- Ironię sytuacyjną:
  - Przykład: Most of us didn't focus in the #ADHD lecture. #irony

Drugi zbiór (w pracy oznaczony jako zbiór B) także pochodzi z Twittera i został zebrany w ramach publikacji zajmującej się analizą różnic między sarkazmem, a ironią [8] , badacze zebrane dane podzielili na 4 zbiory, każdy o zawierający 30 tysięcy próbek, reprezentujące poniższe kategorie:

- Ironiczny
- Sarkastyczny
- Zmieszany zbiór ironiczny i sarkastyczny
- Niezawierający ani ironii, ani sarkazmu

Każdy rekord w ramach zbioru danych posiadał następujące informacje:

- datę opublikowania
- nazwę użytkownika
- unikalny ID Tweetu

Ze względu na to, że wiadomość zawarta w ramach Tweetu nie była podana wprost, tylko sprecyzowany był jej unikalny ID, konieczne było dokonanie dodatkowego mapowania między rekordami, a postami wykorzystując udostępnione przez Tweeter API. Podczas procesu mapowania nie udało się uzyskać wszystkich treści postów, część z nich była nie osiągalna ze względu na różne czynniki losowe takie jak usuniecie konta użytkownika, zablokowanie Tweetu ze względu na naruszenie regulaminu portalu oraz usunięcie Tweetu przez użytkownika. Dlatego uzyskany zbiór postów był mniejszy niż bazowe 30 tysięcy. Na potrzeby pracy zostały wykorzystane 2 zbiory z wcześniej wymienionych:

- Zmieszany zbiór ironiczny i sarkastyczny -> zbiór posiadał 22402 próbek
- Niezawierający ani ironii, ani sarkazmu -> zbiór posiadł 19018 próbek

3. Embeddingi

3.1. Czym jest embedding słów

Embedding słów jest to zbiorcza nazwa na techniki i narzędzia wykorzystane w ramach przetwarzania języka naturalnego pozwalające na dokonanie mapowania słów, ze zbioru znanych pojęć, na wektor liczb rzeczywistych. Z matematycznego punktu widzenia sprowadza się to do transformacji z dyskretnej przestrzeni o wielu wymiarach do ciągłej przestrzeni ze znacznie mniejszą liczbą wymiarów. [9]

Takie podejście powoduje, że możliwe jest badanie podobieństwa słów wykorzystując cosinusowe podobieństwo. Jeśli dwa słowa mają podobne znaczenia lub wykorzystywane są w podobnym kontekście wartość podobieństwa będzie bliższa jednemu. Jeśli słowa rzadko ze sobą występują wartość podobieństwa będzie bliższa zeru. **cooos_def**

3.2. Wykorzystane embeddingi

W ramach pracy są wykorzystywane trzy rodzaje embeddingów, każdy z nich różni się długością wektora i sposobem jego uzyskania. Wykorzystane w pracy embeddingi to:

- Glove
- FastText
- ELMo

3.2.1. Word2vec

Metoda ta nie została wykorzystana w pracy, jednak jest ona kluczowy do zrzumienia podstaw działania i tworzenia ebeddingów dlatego zostanie omówiony w tej sekcji. Występuje w dwóch odmianach:

- CBOW
- Skip grams

Tworzenie embeddingu w oparciu o model typu CBOW polega na trenowaniu prostej sieci neuronowej. Sieć ta składa się z warstwy wejściowej, jednej warstwy ukrytej (warstwy gęstej) oraz warstwy wyjściowej. Trening modelu polega na podawaniu na wejście słów mieszczących się w ramach pewnego założonego okna, przewidując słowo będące w środku tego okna. Jako okno rozumie się tutaj stałą liczbę słów przed i po aktualnie przewidywanym słowem. Słowa, by mogły być przewidywane przez model, są wcześniej konwertowane do przestrzeni wektorowej poprzez wykorzystanie one-hot encodingu. Proces uczenia polega na systematycznym przesuwaniu okna o jedno słowo i trenowanie wag w warstwie ukrytej. Po zakończonym procesie uczenia należy usunąć ostatnią warstwę wyjściową, a wagi z warstwy ukrytej,wykorzystać do uzyskanie embeddingu słów.[10]

Odmiana ‘skip grams’ jest analogiczna do metody CBOW, tylko zamiast przewidywania jednego słowa w kontekście jego otoczenia, przewidywane jest otoczenie w kontekście słowa.

3.2.2. Glove

Metoda opiera na wykorzystaniu globalnych statystyk współwystępowania słów[11], zamiast wykorzystywania tylko informacji o lokalnym kontekście, jak jest to czynione w ramach metody word2vec. Współwystępowanie słów jest zliczane w ramach stałego okna, w ramach które wchodzi skad 10 słów z lewej i z prawej strony.

Trenowanie modelu polega na wyznaczeniu takich embeddingów słów by ich iloczyn skalarny był równy logarytmowi prawdopodobieństwa współwystępowania tych słów w korpusie. Jest to opisane poniższym wzorem:

$$w_i^T \tilde{w}_k = log(P_{ik}) = log(X_{ik}) - log(X_i)$$

gdzie:
w_i - embedding słowa **i**
w_k - embedding słowa **k**
X_{ik} - liczba razy, gdy słowo **k** występuje w kontekście słowa **i**
$P_{ik} = P(k|i) = X_{ik}/X_i$ - prawdopodobieństwo, że słowo **k** wystąpi w kontekście słowa **i**
$X_i = \sum_k X_{ik}$ - liczba razy, gdy jakiekolwiek słowo pojawiło się w kontekście słowa **i**

3.2.3. FastText

Embedding FastText został stworzony w ramach badań prowadzonych przez firmę Facebook[12]. Jest on wariacją metody word2vec, która rozważa zdanie na jeszcze niższym poziomie. Metoda ta rozbija słowa na jeszcze mniejsze fragmenty, tzw. n-gramy. Dla przykładu, angielskie słowo ‘where’ (dla okna o wielkości n=3) jest rozbijane na następujące części: <wh, whe, her, ere, re>. Badacze dodali także specjalne symbole ‘<’ oraz ‘>’ pozwalające poprawnie identyfikować przedrostki i przyrostki. Także by zachować informację o pełnej formie słowa, słowo to jest także dodane do zbioru n-gramów.

Tak uzyskana interpretacja słowa jest konwertowana do postaci numerycznego wektora. Polega to na dodawaniu wektorów kolejnych elementów zbioru n-gramów do jednego długiego wektora. Tak uzyskane wektory poszczególnych słów są podawane analogicznie jak w przypadku tradycyjnego word2vec na wejścia płaskiej sieci neuronowej pozwalając finalnie wyznaczyć embeddingi słów.

3.2.4. ELMo

Ta metoda tworzenia embeddingów nie korzysta już z bezpośrednio z podwalin stworzonych przez metodę word2vec. Zamiast tego do wychwytywania kontekstu słowa, niezbędnego do stworzenia poprawnych embeddingów, wykorzystuje warstwy LSTM [13]. Warstwa ta dzięki swojej budowie pozwala na efektywne zapamiętywanie istotnych informacji, z punktu widzenia zagadnienia klasyfikacji, pojawiających się wcześniej w

ramach sekwencji danych wejściowych. Więcej infrmacji na temat sieci LSTM zawartych jest w rozdziale 5.5.

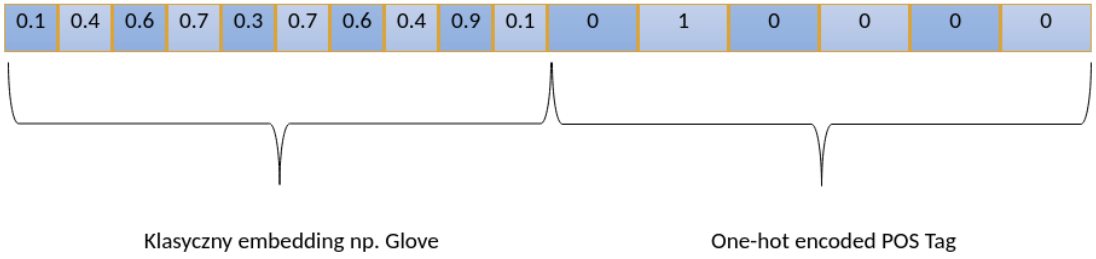
Przetwarzanie zdania wejściowego odbywa się zgodnie z następującymi krokami:

- Zdanie jest dzielone na słowa
- Każde słowo, przy wykorzystaniu sieci pomocniczej typu CNN(opierającej się o warswy konwolucyjne), ma przypisywany embedding. Embedding ten jest określany poprzez agregację wektorowej reprezentacji pojedynczych znaków występujących w ramach słowa. Tak uzyskany embedding jest pierwszym elementem finalnego embeddingu (oznaczony jako E_1)
- Uzyskany wektor jest podawany na wejście pierwszej warstwy Bi-LSTM. Warstwa ta pozwala na analizę zdania zarówno od początku do końca, jak i od końca do początku.
- Stany ukryte z pierwszej warstwy dla każdego słowa są agregowane w jeden wektor i stanowią drugi element finalnego embeddingu (oznaczony jako E_2)
- Następnie dane o stanie ukrytym pierwszej warstwy są przekazywane na drugą warstwę Bi-LSTM
- Stany ukryte z drugiej warstwy dla każdego słowa są agregowane w jeden wektor i stanowią drugi element finalnego embeddingu (oznaczony jako E_3)
- W ostaniem kroku następuje połącznie uzyskanych fragmentów embeddingów, oznaczonych jako E_1, E_2, E_3 w jeden o standardowym wymiarze 1024

3.3. Uwzględnienie POS tagów w ramach embeddingu

POS tag (Part-of-speach tag) zawiera informacje na temat cech morfosyntatycznych słowa [14].Najbardziej podstawową kategorią są powszechnie znane części mowy, takie jak czasownik, rzeczownik, przymiotnik. Jednak nie jest to jedyna informacja zawarta w ramach takiego tagu, przechowuje on bowiem także informację o czasie w jakim jest użyty czasownik, w której liczbie jest wykorzystany rzeczownik i wiele innych.

Aby umieścić informacje na temat części mowy tak by była ona możliwa do interpretacji w ramach embeddingu konieczne jest konwertowanie jej do postaci wektorowej. Ze względu na brak dużego podobieństwa między poszczególnymi częściami mowy zdecydowano się na zakodowanie tej informacji w formie one-hot encoding. Tak utworzony wektor był doklejany do istniejącego już embeddingu uzyskanego innymi metodami tworząc wektor dłuższy o 46 elementów.



Rysunek 1. Przykład embeddingu wraz z zakodowaną informacją o POS tagu

4. Preprocessing

Aby zapewnić jak najlepszą jakość klasyfikacji konieczne jest obrobienie danych do postaci pozbawionej zbędnych szumów. Na potrzeby tej pracy zostały podjęte następujące kroki obróbki danych:

1. Zastąpienie linków tagiem “<url>”.
2. Usunięcie znaków końca linii.
3. Zastąpienie wystąpień nazw użytkowników tagiem “<username>”.
4. Zamiana emotikon z formatu “:XX_YY:” na format “<emote> XX YY” .
5. Rozbicie hasztagów na mniejsze słowa, oznaczenie go tagiem “<hashtag>”.
6. Zastąpienie oznaczeń odnoszących się do czasu tagiem “<time>”.
7. Zastąpienie liczb tagiem “<number>”.
8. Usunięcie znaków ” | ” oraz wielokropków.
9. Konwersja do małych liter.
10. Rozwinięcie form skróconych do ich pełnych form.

Tabela 1. Tabela przedstawiająca efekt preprocessingu.

Numer operacji preprocessingu	Przed konwersją	Po konwersji
1	www.google.pl	<url>
2	\n \r \n	
3	@mike	<username>
4	:face_screaming_in_fear:	<emote> face screaming in fear
5	#trueFriend	<hashtag> true Friend
6	14:58	<time>
7	1234	<number>
8	|...	.
9	Kazimierz Wielki	kazimierz wielki
10	I’ll | He’s	I will | He has

5. Warstwy

5.1. DENSE

Jest to najbardziej podstawowy typ warstwy, w której każdy neuron uzyskuje informacje ze wszystkich wejść. Następnie agreguje te dane, przy uwzględnieniu wag odpowiadającym wejściom i przy wykorzystaniu swojej funkcji aktywacji wylicza wartość wyjściową neuronu.

Często jedna warstwa gęsto połączona nie wystarcza by poprawnie rozwiązywać dane zadanie klasyfikacji, dlatego wykorzystuje się kilka takich warstw ze sobą połączonych tak, że wyjścia jednej warstwy są jednocześnie wejściami kolejnej warstwy. Należy jednak pamiętać, że funkcja aktywacji musi być wtedy funkcją nieliniową. Obecnie najczęściej wykorzystywaną funkcją aktywacji jest funkcja ReLU opisana poniższym wzorem:

$$f(x) = max(0,x) = \begin{cases} 0, \; dla \; x \leq 0 \\ x, \; dla \; x > 0 \end{cases}$$

5.2. Warstwa konwolucyjna

Sieci używające warstw konwolucyjnych (takie sieci oznaczane są często jako CNN) z reguły wykorzystywane są w ramach analizy obrazu. Działają one w oparciu o różne filtry przemieszczające się systematycznie po macierzy obrazu. Dany obraz może być jednocześnie analizowany przez wiele filtrów przemieszczających się z różnym skokiem. Pozwala to na naukę rozpoznawania różnych cech obrazu umożliwiających zagadnienia klasyfikacji. Kolejne warstwy konwolucyjne można ze sobą łączyć pozwalając na rozpoznawanie coraz bardziej zaawansowanych cech obrazu.

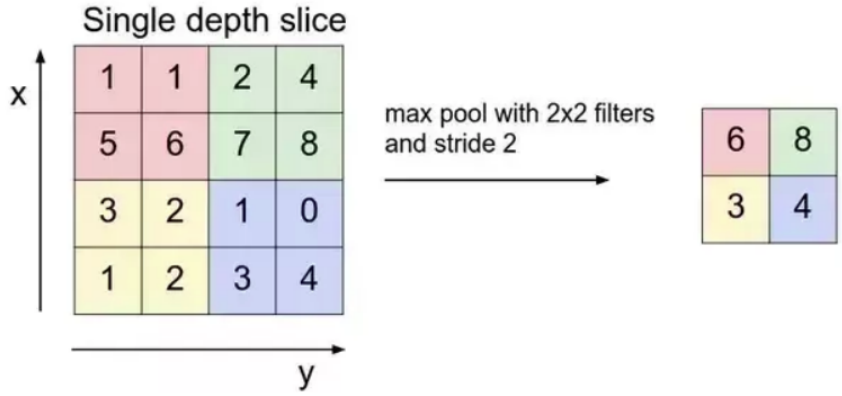
W przypadku analizy tekstu macierzą wejściową dla warstwy konwolucyjnej jest macierz o wymiarze AxB, gdzie A to liczba słów w zdaniu, a B to rozmiar embeddingu. Natomiast, ze względu na strukturę danych, sama ekstrakcja cech dokonywana przez filtr może przebiegać tylko w ramach wymiaru A, gdyż dane wzdłuż wymiaru B reprezentują unikalne cechy danego słowa i nie powinny być agregowane.

5.3. Pooling

Zadaniem warstwy poolingu jest zmniejszenie wymiarowości macierzy uzyskanej z poprzedniej warstwy (bardzo często występuję tuż po warstwie konwolucyjnej). Można ten proces porównać do przeprowadzania operacji zmniejszania rozdzielczości obrazu. Operacja ta polega na systematycznym przesuwaniu stałego okna po macierzy wejściowej i dokonywania pewnej formy agregacji danych (wybór elementu o najwyższej wartości, wyliczenie średniej z wartości elementów) na elementach macierzy zawierających się w bieżącym oknie. Dzięki takiemu zabiegowi kolejne warstwy sieci mogą skupiać się na

analizie i wykrywaniu bardziej skomplikowanych wzorców w oparciu o cechy uwypuklone przez warstwę poolingu.

Najbardziej popularną formą poolingu jest max pooling, który polega na wybieraniu elementu o najwyższej wartości spośród elementów w oknie. Na rysunku XXX przedstawiono przykład takiej operacji przy zastosowaniu okna o rozmiarze 2x2 i przesunięciu wynoszącym 2.



Rysunek 2. Wynik operacji max poolingu

5.4. RNN

Neuronowa sieć rekurencyjna (RNN) powstała z myślą o przewidywaniu kolejnych elementów sekwencji. Jako wejście przyjmuje kolejne elementy sekwencji, ale w przeciwieństwie do sieci typu feedforward posiada także swój wewnętrzny stan, w ramach którego zawarta jest informacja o poprzednich wejściach. Sieć tego typu charakteryzuje się jednak bardzo istotną wadą, mianowicie boryka się ona z problem zanikającego gradientu. Jest on spowodowany naturą propagacji wstecznej i wiąże się z coraz mniejszym wpływem początkowych elementów sekwencji na wyjście modelu. Aby temu zaradzić zostały wprowadzone nowe architektury warstw, jedną z nich jest warstwa typu LSTM.

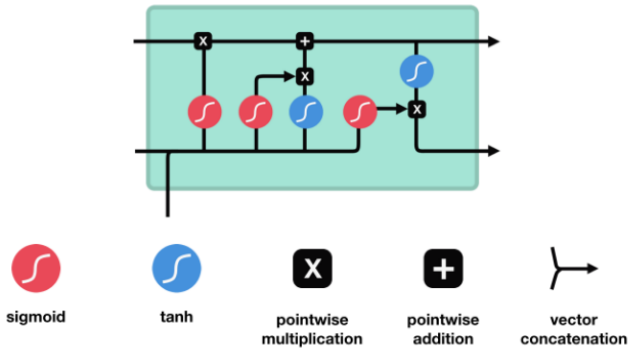
5.5. LSTM

Warstwa typu LSTM składa się nuronów zawierających w sobie wiele pośrednich operacji. Poniżej zostaną omówione najważniejsze z nich.

5.5.1. Funkcje aktywacji

Neuron LSTM wykorzystuje dwie funkcje aktywacji:

- Tahn (funkcja aktywacji typu tangens hiperboliczny) - pozwala ona na normalizowanie wektora tak by wartości zawierały się w ramach przedziału [-1,1]
- Sigmoid (funkcja aktywacji sigmoidalna) - przekształca wartości wektora do wartości z przedziału [0,1]. Istotna z punktu widzenia określania, które elementy wektora



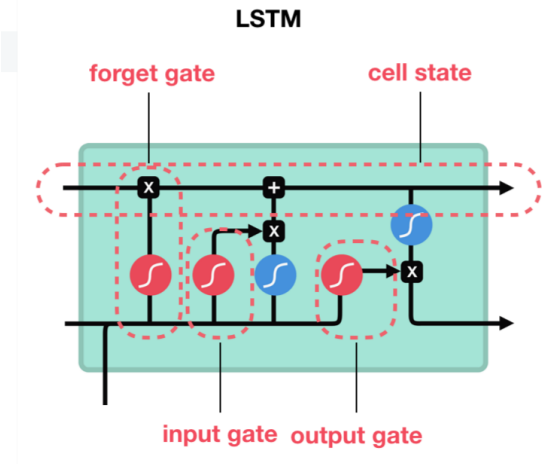
Rysunek 3. Przepływ danych w pojedynczym neuronie LSTM

powinny zostać zapamiętane (wartość bliżej 1), a które zapomniane (wartość bliżej 0)

5.5.2. Elementy neuronu

W skład neuronu LSTM wchodzą następujące elementy:

- Forget gate
- Input gate
- Cell state
- Output gate



Rysunek 4. Elementy wewnątrz neuronu LSTM

Forget gate

Zadaniem tej bramy jest decydowanie jaka informacja powinna zostać zachowana przez neuron, a jaka usunięta. Informacje z poprzedniego stanu ukrytego (h_{t-1}) i informacje z bieżącego wejścia(x_t) są przekazywane do sigmoidalnej funkcji aktywacji. Powstała macierz jest oznaczona jako f_t. Dla każdej pozycji w macierzy zwracane są wartości z

zakresu [0,1]. Im bliżej wartości zero, tym wpływ danej pozycji zanika, im bliżej wartości jeden tym wpływ danej pozycji rośnie.

Input gate

Służy do aktualizacji wewnętrznego stanu neuronu ('cell state'). Informacje z poprzedniego stanu ukrytego (h_{t-1}) i informacje z bieżącego wejścia(x_t) są przekazywane do sigmoidalnej funkcji aktywacji. Tak jak w przypadku 'forget gate' uzyskiwana jest macierz o wartościach z zakresu [0,1], która tym razem określa jak bardzo informacja, powstała z połącznia h_{t-1} oraz x_t, jest ważna z punktu widzenia zadania klasyfikacji. Wyjście z tej operacji oznaczone jest jako i_t.

W ramach tej bramy dokonuje się normalizacja informacji zawartych w ramach połączenia h_{t-1} oraz x_t, poprzez wykorzystanie tangensa hiperbolicznego. Wyjście z tej operacji oznaczone jest jako c_t.

Kolejnym krokiem jest agregacja informacji z i_t oraz c_t poprzez wykonanie operacji mnożenia macierzy.

Cell state

Ten element neuronu odpowiada za jego stan wewnętrzny i jest przekazywany między kolejnymi komórkami w warstwie. Stan uzyskany z poprzedniej komórki oznaczony jest jako c_{t-1}, a z bieżącej jako c_t. W celu uzyskania bieżącego stanu komórki wykorzystywane są informacje zgromadzone w ramach poprzednich bram. W pierwszej kolejności dokonywane jest mnożenie macierzowe c_{t-1} oraz f_t, które pozwala na usunięcie elementów macierzy nieistotnych z punktu widzenia sieci. Następnie dokonywana jest operacja dodawania uzyskanej macierzy z macierzą uzyskaną w ramach 'input gate'. Powoduje to dodanie do stanu neuronu nowych informacji istotnych dla sieci. Stan po tej operacji jest określany zmienną c_t.

Output gate

Zadaniem tej bramy jest określenie jak powinien wyglądać stan ukryty przekazywany do kolejnego neuronu. Stan ten zawiera informacje na temat poprzednich wejść istotnych z dla zagadnienia klasyfikacji. W ramach tej bramy dokonywane są trzy operacje. Pierwszą z nich jest podanie zagregowanych informacji z poprzedniego stanu ukrytego (h_{t-1}) i informacje z bieżącego wejścia(x_t) na sigmoidalną funkcję aktywacji. W ramach drugiej operacji wewnętrzy stan neuronu ('cell state') jest normalizowany przy wykorzystaniu tangesna hiperbolicznego. Trzecią operacją jest mnożenie macierzowe wyników dwóch poprzednich operacji, w efekcie uzyskiwany jest stan ukryty neuronu, oznaczony jako h_t, przekazywany do następnego neuronu.

5.6. Bi-LSTM

Warstwa składa się z dwóch tak samo zbudowanych warstw LSTM. Jedna z warstw przetwarza ciąg danych wejściowych od początku do końca, a druga od końca do początku. Pozwala to na naukę zależności między elementami w dwóch kierunkach, co przekłada się na większą liczbę informacji na temat sekwencji, co pozwala na szybsze i dokładniejsze trenowanie modelu.

6. Metryki

Jakość klasyfikacji można monitorować przy wykorzystaniu różnych metryk. Najbardziej popularną metryką jest dokładność (ang. accuraccy), czyli stosunek poprawnie zakwalifikowanych przez model obserwacji, do całkowitej liczby obserwacji w zbiorze. Nie jest to jednak miara idealna, nie bierze ona pod uwagę między innymi tego czy badany zbiór jest zbilansowany. W przypadku zbioru niezbilansowanego możliwe jest uzyskanie bardzo dobrej dokładności nawet przy zakwalifikowaniu wszystkich obserwacji do tylko jednej klasy. W celu wyeliminowania tego problemu wykorzystuje się inne metryki, takie jak precyzja i czułość.

Powyższe metryki wykorzystują następujące pojęcia pomocnicze:

- true positive (TP) - liczba pozytywnych obserwacji zaklasyfikowanych jako pozytywne
- true negative (TN) - liczba negatywnych obserwacji zaklasyfikowanych jako negatywne
- false positive (FP) - liczba negatywnych obserwacji zaklasyfikowanych jako pozytywne
- false negative (FN) - liczba pozytywnych obserwacji zaklasyfikowanych jako negatywne

W oparciu o te pojęcia pomocnicze można sformułować następujące zależności:

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

$$czułość = \frac{TP}{TP + FN}$$

W oparciu o wzory można zauważyć, że precyzja zawiera w sobie informację jaka część obserwacji zakwalifikowanych jako pozytywne była zakwalifikowana poprawnie. Natomiast czułość zawiera informację na temat tego jaka część pozytywnych obserwacji została zakwalifikowana poprawnie.

Precyzja i czułość są ze sobą współzależne, w przypadku, gdy wskazania jednej z metryk się poprawiają, to wskazania drugiej metryki spadają. Aby monitować optymalne wskazania obu metryk powstała metryka F1, jest ona opisana poniższym wzorem:

$$F1 = 2 \cdot \frac{precyzja \cdot czułość}{precyzja + czułość}$$

7. Eksperymenty

Na potrzeby analizy wykorzystano wiele różnych modeli sieci neuronowych, bazujących głownie na trzech typach warstw:

- gęstej;
- konwolucyjnej;
- LSTM;

W ramach różnych architektur testowany był wpływ kilku hiperparametrów na jakość klasyfikacji. Oto lista niektórych z nich:

- dla warstwy gęstej:
  - 1/2/3 warstwy
  - Z dropoutem / bez dropoutu
  - Wielkość wartwy ukrytej 30-200
- dla warstwy konwolucyjnej;
  - 1/2/3 warstwy
  - Wielkość okna 3-15
  - Z dropoutem / bez dropoutu
  - Wielkość wartwy ukrytej 30-200
- dla warstwy LSTM;
  - Wykorzystanie warstwy pojedynczej (LSTM) oraz podwójnej (Bi-LSTM)
  - 1/2/3 warstwy
  - Z dropoutem / bez dropoutu
  - Wielkość wartwy ukrytej 30-200

7.1. Wyniki analiza

7.1.1. Eksperyment 1 - Analiza wpływu POS tagów na jakoś klasyfikacji

Aby w pełni zweryfikować wpływ części mowy (POS tagów) na jakość klasyfikacji zostały wykorzystane trzy różne embeddingi oraz dwa różne zbiory danych wejściowych. Wykorzsytane embeddingi to: Glove, FastText oraz ELMo, a ich dokładniejszy opis znajduje się w rozdziale 3. Natomiast dwa zbiory danych wejściowych (oznaczonych jako A oraz B) są opisane w rozdziale 2. Na potrzeby budowania modeli każdy ze zbiorów danych został podzielony na trzy podzbiory: treningowy, walidacyjny oraz testowy w proporcjach 60/20/20 procent.

Wyniki uzyskane przez modele na danych testowych zbioru A zostały umieszczone w rozdziale 7.2.1 i 7.2.2 w tabelach 2, 3, 4 , a dla zbioru B w tabelach 5, 6, 7.

W oparciu o uzyskane wyniki można stwierdzić, że wraz z dołączeniem informacji o częściach mowy nie obserwuje się widocznej poprawy jakości klasyfikacji, czasem występuje nawet jej pogorszenie. Jednym z powodów takiego zachowania może być fakt,

że rodzaj części mowy nie jest kluczowy do rozpoznania ironicznego charakteru tekstu. Możne to być też spowodowane tym, że sieci wychwytują niezbędne dla nich informacje na temat roli konkretnego słowa w zdaniu w oparciu o uzyskiwany kontekst i dodatkowa informacja w ramach embeddingu jest już niepotrzebna.

Ponadto można zaważyć, że wraz ze wzrostem rozmiaru embedingów poprawia się jakość klasyfikacji, co jest zgodne z oczekiwaniami, jako że dłuższy embedding pozwala na zachowanie większej liczby informacji na temat słowa.

Analizując uśrednione pomiary jakości klasyfikacji dla sieci opartych o różne typy warstw, można zauważyć, że najgorzej sprawują się sieci oparte o warstwy gęste. Natomiast sieci wykorzystujące warstwy konwolucyjne oraz LSTM posiadają podobną skuteczność klasyfikacji. Aby wykluczyć wpływ zaszumienia danych oraz przeuczenia sieci wykonano jeszcze dwa eksperymenty mające porównać skuteczność klasyfikacji sieci opartych na warstwach konwolucyjnych oraz LSTM. Eksperymenty te są przeprowadzone na danych wejściowych pozbawionych informacji o POS tagach, gdyż z dotychczasowych pomiarów wynika, iż mają one pomijalny wpływ na jakość klasyfikacji.

7.1.2. Eksperyment 2 - Uczenie modelu na zbiorze B i testowanie jakości klasyfikacji na zbiorze A

W ramach tego eksperymentu zbiór danych oznaczonych jako **B** został podzielony na dwa zbiory: treningowy oraz w walidacyjny w proporcjach 80/20 procent. Natomiast zbiór oznaczony jako A został w całości wykorzystany jako zbiór testowy.

Wyniki uzyskane przez modele w ramach tego eksperymentu zostały umieszczone w rozdziale 7.2.3 w tabelach 8, 9, 10.

W oparciu o uzyskane miary jakości klasyfikacji można stwierdzić, że modele wyuczone na zbiorze B źle radzą sobie z klasyfikacją rekordów ze zbioru A. Dokładność na poziomie pięćdziesięciu paru procent wskazuje, iż pomimo tego, że oba zbiory pochodzą z tego samego źródła to różnią się one na tyle bardzo swoją strukturą wewnętrzną, że żadna architektura nie może sobie poradzić z prawidłową klasyfikacją. Co potwierdza powszechnie znaną prawdę dotyczącą sieci neuronowych, że takie modele do procesu trenowania potrzebują bardzo dużej ilości danych o możliwie jak najbardziej zróżnicowanej strukturze. Wystarczy, że zbiór danych treningowych jest tylko pewnym wycinkiem dziedziny problemu, by sieć podczas treningu nauczyła się klasyfikacji w opraciu o złe lub nieistotne cechy.

7.1.3. Eksperyment 3 - Uczenie i testowanie modelu na fragmencie zbioru B tak by był równoliczny ze zbiorem A

W ramach tego eksperymentu ze zbioru danych oznaczonych jako **B** zostało wybranych w losowy sposób 2000 rekordów ironicznych i tyle samo nieironicznych, tak by licznością były porównywalne ze zbiorem oznaczonym jako **A**. Tak uzyskany zbiór został podzielony na trzy podzbiory: treningowy, walidacyjny oraz testowy w proporcjach 60/20/20 procent.

Wyniki uzyskane przez modele w ramach tego eksperymentu zostały umieszczone w rozdziale 7.2.4 w tabelach 11, 12, 13.

W opraciu o uzyskane wyniki można zauważyć, dużo wyższą jakość klasyfikacji w porównaniu do zbioru oznaczonego jako **A**. Świdczyć to może o bardzo małej różnorodności wśród danych wewnątrz zbioru **B**. Najprawdopodobniej wszystkie elementy tego zbioru reprezenntują ten sam najprostszy typ ironii bazujący na przeciwnej polaryzacji w ramach ironii werbalnej.

Natomiast w kwestii porównania jakości klasyfikacji sieci CNN i sieci opratych o wartwy LSTM, trudno jest dostrzec zauważalną różnicę w jakościach klasyfikacji przez sieci o najwyższych wynikach. Obie architektury sprawiają się równie dobrze w przypadku klasyfikacji na zbiorze B, nawet przy dość małej ilości danych.

7.2. Wyniki eksperymentów

W ramach tego rozdziału umieszczone są dane dotyczące jakości klasyfikacji poszczególnych modeli w ramach wykonanych eksperymentów. Dla eksperymentu pierwszego dla każdej metryki zostało dokonane porównanie wartości dla modelu trenowego z inforamcją o POS tagach (kolumna oznaczona jako P) z modelem trenowanym bez inforamcji o POS tagach (kolumna oznaczona jako N). Dla każdej grupy pomiarów kolorem zielonym została zaznaczona największa wartość danej metrki.

7.2.1. Wyniki dla eksperymentu 1 - zbiór A

Tabela 2. Embedding Glove o długości 25, zbiór A

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	0.6327	**0.6372**	0.6074	**0.6210**	**0.6853**	0.6456	**0.6440**	0.6331
dense_2	**0.6022**	0.5943	**0.5790**	0.5732	**0.6573**	0.6386	**0.6157**	0.6041
dense_3	**0.6192**	0.6079	**0.5946**	0.5748	0.6736	**0.7342**	0.6316	**0.6448**
dense_4	**0.6395**	0.6169	**0.6095**	0.5978	**0.7132**	0.6410	**0.6573**	0.6186
lstm_1	0.6440	**0.6677**	**0.6446**	0.6261	0.5920	**0.7808**	0.6172	**0.6950**
lstm_2	**0.6655**	0.6271	**0.6394**	0.5932	0.7109	**0.7342**	**0.6732**	0.6562
lstm_3	**0.6632**	0.6598	0.6276	**0.6624**	**0.7505**	0.6083	**0.6836**	0.6342
lstm_4	0.6598	**0.6644**	0.6151	**0.6325**	**0.7972**	0.7342	**0.6944**	0.6796
lstm_5	**0.6711**	0.6587	**0.6347**	0.6191	0.7575	**0.7692**	**0.6907**	0.6860
lstm_6	**0.6677**	0.6632	0.6470	**0.6541**	**0.6923**	0.6480	**0.6689**	0.6510
lstm_7	**0.6700**	0.6531	**0.6395**	0.6150	0.7319	**0.7599**	**0.6826**	0.6798
lstm_8	0.6429	**0.6474**	**0.6506**	0.6291	0.5687	**0.6643**	0.6069	**0.6462**
lstm_9	0.6485	**0.6644**	0.6006	**0.6823**	**0.8205**	0.5757	**0.6935**	0.6245
lstm_10	**0.6644**	0.6531	**0.6460**	0.6150	0.6806	**0.7599**	0.6628	**0.6798**
lstm_11	0.6361	**0.6610**	**0.6389**	0.6267	0.5734	**0.7435**	0.6044	**0.6801**
lstm_12	0.6689	**0.6689**	**0.6497**	0.6387	0.6876	**0.7296**	0.6681	**0.6811**
lstm_13	**0.6519**	0.6497	0.6170	**0.6178**	**0.7435**	0.7272	**0.6744**	0.6680
lstm_14	0.6610	**0.6632**	**0.6532**	0.6367	0.6410	**0.7109**	0.6470	**0.6718**
lstm_15	0.6576	**0.6598**	0.6211	**0.6385**	**0.7529**	0.6876	**0.6807**	0.6621
lstm_16	0.6677	**0.6677**	0.6464	**0.6496**	**0.6946**	0.6829	**0.6696**	0.6659
cnn_1	**0.6519**	0.6485	**0.6416**	0.6189	0.6386	**0.7156**	0.6401	**0.6637**
cnn_2	0.6316	**0.6485**	0.6107	**0.6365**	**0.6620**	0.6410	0.6353	**0.6387**
cnn_3	0.6203	**0.6384**	0.5954	**0.6326**	**0.6759**	0.6060	**0.6331**	0.6190
cnn_4	0.6440	**0.6508**	**0.6515**	0.6345	0.5710	**0.6596**	0.6086	**0.6468**
cnn_5	0.6327	**0.6395**	0.6125	**0.6200**	0.6596	**0.6620**	0.6352	**0.6403**
cnn_6	0.6395	**0.6598**	0.6238	**0.6422**	0.6456	**0.6736**	0.6345	**0.6575**
cnn_7	0.6225	**0.6237**	**0.5975**	0.5810	0.6783	**0.8018**	0.6353	**0.6738**
cnn_8	0.6305	**0.6485**	0.6153	**0.6347**	0.6340	**0.6480**	0.6245	**0.6412**
cnn_9	**0.6406**	0.6338	**0.6241**	0.6143	0.6503	**0.6573**	**0.6369**	0.6351
cnn_10	0.6293	**0.6440**	0.6150	**0.6350**	**0.6293**	0.6247	0.6221	**0.6298**
cnn_11	**0.6395**	0.6327	**0.6227**	0.6111	0.6503	**0.6666**	0.6362	**0.6376**

Tabela 3. Embedding FastText o długości 300, zbiór A

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	**0.6598**	0.6497	**0.6355**	0.6307	**0.6993**	0.6689	**0.6659**	0.6493
dense_2	**0.6677**	0.6576	**0.6490**	0.6334	0.6853	**0.6969**	**0.6666**	0.6637
dense_3	**0.6440**	0.6327	**0.6126**	0.6	0.7226	**0.7272**	**0.6631**	0.6575
dense_4	**0.6655**	0.6418	**0.6501**	0.6129	0.6713	**0.7086**	**0.6605**	0.6572
lstm_1	0.6813	**0.6824**	**0.6909**	0.6439	0.6200	**0.7715**	0.6535	**0.7020**
lstm_2	0.6485	**0.6655**	0.6322	**0.6405**	0.6573	**0.7062**	0.6445	**0.6718**
lstm_3	0.6632	**0.6813**	0.6219	**0.6622**	**0.7785**	0.6993	**0.6915**	0.6802
lstm_4	**0.6779**	0.6576	**0.6463**	0.6184	0.7412	**0.7668**	**0.6905**	0.6847
lstm_5	**0.6870**	0.6677	**0.6681**	0.6537	**0.7039**	0.6689	**0.6855**	0.6612
lstm_6	**0.6926**	0.6813	**0.6780**	0.6496	0.6969	**0.7435**	0.6873	**0.6934**
lstm_7	**0.6813**	0.6553	**0.6689**	0.6152	0.6783	**0.7715**	0.6736	**0.6845**
lstm_8	**0.6711**	0.6700	0.6431	**0.6751**	**0.7226**	0.6153	**0.6805**	0.6439
lstm_9	**0.6779**	0.6497	**0.6698**	0.6064	0.6620	**0.7902**	0.6658	**0.6862**
lstm_10	**0.6836**	0.6474	**0.6769**	0.6338	**0.6643**	0.6456	**0.6705**	0.6397
lstm_11	**0.6542**	0.6508	**0.6242**	0.6162	0.7202	**0.7412**	0.6688	**0.6730**
lstm_12	**0.6836**	0.6768	**0.6712**	0.6682	**0.6806**	0.6620	**0.6759**	0.6651
lstm_13	**0.6779**	0.6757	0.6538	**0.6748**	**0.7132**	0.6386	**0.6822**	0.6562
lstm_14	0.6711	**0.6937**	0.6526	**0.7219**	**0.6876**	0.5990	**0.6696**	0.6547
lstm_15	**0.6790**	0.6508	**0.6593**	0.6102	0.6993	**0.7738**	0.6787	**0.6824**
lstm_16	**0.6824**	0.6757	0.6674	**0.6961**	**0.6876**	0.5874	**0.6773**	0.6371
cnn_1	**0.6564**	0.6384	**0.6310**	0.6142	**0.7016**	0.6829	**0.6644**	0.6467
cnn_2	**0.6542**	0.6429	**0.6413**	0.6263	0.6503	**0.6526**	**0.6458**	0.6392
cnn_3	0.6271	**0.6429**	0.6092	**0.6287**	0.6433	**0.6433**	0.6258	**0.6359**
cnn_4	0.6372	**0.6474**	**0.6310**	0.6258	0.6060	**0.6783**	0.6183	**0.6510**
cnn_5	**0.6598**	0.6542	**0.6428**	0.6382	**0.6713**	0.6620	**0.6567**	0.6498
cnn_6	0.6485	**0.6519**	0.6213	**0.6257**	**0.7039**	0.7016	0.6601	**0.6615**
cnn_7	0.6644	**0.6644**	0.6392	**0.6434**	**0.7062**	0.6899	**0.6710**	0.6659
cnn_8	**0.6451**	0.6395	**0.6220**	0.6211	**0.6829**	0.6573	**0.6511**	0.6387
cnn_9	0.6372	**0.6485**	0.6168	**0.6288**	0.6643	**0.6713**	0.6397	**0.6493**
cnn_10	0.6485	**0.6485**	0.6277	**0.6299**	**0.6759**	0.6666	**0.6509**	0.6477
cnn_11	0.6451	**0.6542**	0.6321	**0.6363**	0.6410	**0.6689**	0.6365	**0.6522**

Tabela 4. Embedding ELMo o długości 1024, zbiór A

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	0.6531	**0.6779**	0.6392	**0.6487**	0.6526	**0.7319**	0.6459	**0.6878**
dense_2	**0.6519**	0.6429	**0.6273**	0.6184	**0.6946**	0.6876	**0.6592**	0.6512
dense_3	0.6203	**0.6418**	0.5882	**0.6261**	**0.7226**	0.6480	**0.6485**	0.6368
dense_4	0.6406	**0.6485**	0.6065	**0.6520**	**0.7365**	0.5897	**0.6652**	0.6193
lstm_1	**0.6836**	0.6655	**0.6795**	0.6164	0.6573	**0.8205**	0.6682	**0.704**
lstm_2	**0.7152**	0.6903	**0.7015**	0.6611	0.7179	**0.7412**	**0.7096**	0.6989
lstm_3	**0.6926**	0.6711	**0.6891**	0.6443	0.6666	**0.7179**	0.6777	**0.6791**
lstm_4	0.6802	**0.6892**	**0.6666**	0.6492	0.6806	**0.7808**	0.6735	**0.7089**
lstm_5	**0.6971**	0.6768	**0.6687**	0.6382	0.7435	**0.7692**	**0.7041**	0.6976
lstm_6	0.6926	**0.7129**	0.6772	**0.6826**	0.6993	**0.7622**	0.6880	**0.7202**
lstm_7	**0.7186**	0.7118	**0.7**	0.6986	**0.7342**	0.7132	**0.7167**	0.7058
lstm_8	0.6734	**0.6926**	0.6933	**0.7524**	**0.5850**	0.5454	**0.6346**	0.6324
lstm_9	0.6903	**0.7050**	0.6962	**0.7038**	0.6410	**0.6759**	0.6674	**0.6896**
lstm_10	**0.6892**	0.6847	**0.6824**	0.6820	**0.6713**	0.6550	**0.6768**	0.6682
lstm_11	0.6757	**0.6802**	**0.6658**	0.6442	0.6643	**0.7599**	0.6651	**0.6973**
lstm_12	**0.6971**	0.6813	0.6716	**0.6981**	**0.7342**	0.6037	**0.7015**	0.6475
lstm_13	**0.7039**	0.6711	**0.6937**	0.6699	**0.6969**	0.6340	**0.6953**	0.6514
lstm_14	**0.6960**	0.6903	0.66	**0.7066**	**0.7692**	0.6177	**0.7104**	0.6592
lstm_15	**0.6949**	0.6892	0.6593	**0.6816**	**0.7668**	0.6736	**0.7090**	0.6776
lstm_16	0.6937	**0.7016**	**0.7078**	0.6813	0.6270	**0.7226**	0.6650	**0.7013**
cnn_1	0.6282	**0.6418**	0.6086	**0.6296**	**0.6526**	0.6340	0.6299	**0.6318**
cnn_2	0.6508	**0.6531**	**0.6333**	0.6326	0.6643	**0.6783**	0.6484	**0.6546**
cnn_3	**0.6497**	0.6485	**0.64**	0.6340	0.6340	**0.6503**	0.6370	**0.6421**
cnn_4	**0.6564**	0.6542	**0.6443**	0.6394	0.6503	**0.6573**	0.6473	**0.6482**
cnn_5	**0.6598**	0.6587	**0.6454**	0.6420	0.6620	**0.6689**	0.6536	**0.6552**
cnn_6	**0.6519**	0.6203	**0.6437**	0.6120	**0.6317**	0.5920	**0.6376**	0.6018
cnn_7	0.6610	**0.6632**	**0.6592**	0.6356	0.6223	**0.7156**	0.6402	**0.6732**
cnn_8	**0.6451**	0.6406	**0.6258**	0.6158	0.6666	**0.6876**	0.6455	**0.6497**
cnn_9	**0.6519**	0.6384	0.6318	**0.6339**	**0.6759**	0.6013	**0.6531**	0.6172
cnn_10	**0.6632**	0.6497	0.6427	**0.6461**	**0.6876**	0.6130	**0.6644**	0.6291
cnn_11	0.6293	**0.6451**	0.5980	**0.6125**	0.7179	**0.7296**	0.6525	**0.6659**

7.2.2. Wyniki dla eksperymentu 1 - zbiór B

Tabela 5. Embedding Glove o długości 25, zbiór B

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	0.8477	**0.8593**	0.8460	**0.8503**	0.8526	**0.8744**	0.8493	**0.8622**
dense_2	0.8483	**0.8613**	0.8349	**0.8551**	0.8709	**0.8724**	0.8525	**0.8636**
dense_3	0.8433	**0.8604**	0.8403	**0.8510**	0.8504	**0.8762**	0.8453	**0.8634**
dense_4	0.8493	**0.8646**	0.8455	**0.8731**	**0.8574**	0.8554	0.8514	**0.8641**
lstm_1	0.9914	**0.9921**	0.9992	**0.9997**	0.9837	**0.9847**	0.9914	**0.9921**
lstm_2	**0.9910**	0.9909	0.9984	**0.9989**	**0.9837**	0.9829	**0.9910**	0.9909
lstm_3	0.9899	**0.9910**	0.9966	**0.9987**	0.9832	**0.9834**	0.9899	**0.9910**
lstm_4	**0.9910**	0.9906	0.9977	**0.9982**	**0.9844**	0.9832	**0.9910**	0.9906
lstm_5	0.9906	**0.9906**	**0.9989**	0.9974	0.9824	**0.9839**	0.9906	**0.9906**
lstm_6	**0.9911**	0.9895	**0.9972**	0.9969	**0.9852**	0.9822	**0.9911**	0.9895
lstm_7	**0.9912**	0.9909	0.9987	**0.9992**	**0.9839**	0.9827	**0.9912**	0.9909
lstm_8	0.9918	**0.9918**	**0.9989**	0.9987	0.9847	**0.9849**	0.9917	**0.9918**
lstm_9	**0.9915**	0.9914	0.9974	**1**	**0.9857**	0.9829	**0.9915**	0.9914
lstm_10	**0.9925**	0.9918	0.9994	**0.9997**	**0.9857**	0.9839	**0.9925**	0.9917
lstm_11	**0.9918**	0.9914	0.9982	**0.9982**	**0.9854**	0.9847	**0.9918**	0.9914
lstm_12	**0.9923**	0.9916	**0.9994**	0.9977	0.9852	**0.9857**	**0.9923**	0.9916
lstm_13	**0.9924**	0.9916	**0.9997**	0.9992	**0.9852**	0.9842	**0.9924**	0.9916
lstm_14	**0.9916**	0.9911	0.9994	**0.9994**	**0.9839**	0.9829	**0.9916**	0.9911
lstm_15	0.9902	**0.9909**	0.9974	**0.9989**	**0.9832**	0.9829	0.9902	**0.9909**
lstm_16	**0.9923**	0.9907	**0.9997**	0.9984	**0.9849**	0.9832	**0.9923**	0.9907
cnn_1	0.9621	**0.9680**	0.9723	**0.9814**	0.9518	**0.9546**	0.9620	**0.9678**
cnn_2	0.9704	**0.9706**	0.9847	**0.9850**	0.9561	**0.9561**	0.9702	**0.9703**
cnn_3	0.9452	**0.9471**	0.9630	**0.9704**	**0.9268**	0.9230	0.9445	**0.9461**
cnn_4	0.9501	**0.9538**	0.9701	**0.9733**	0.9295	**0.9338**	0.9494	**0.9531**
cnn_5	**0.9569**	0.9563	**0.9833**	0.9732	0.9303	**0.9391**	**0.9560**	0.9558
cnn_6	**0.9919**	0.9918	**0.9994**	0.9984	0.9844	**0.9852**	**0.9919**	0.9918
cnn_7	**0.9924**	0.9921	**0.9997**	0.9992	0.9852	**0.9852**	**0.9924**	0.9921
cnn_8	0.9603	**0.9656**	0.9796	**0.9796**	0.9408	**0.9516**	0.9598	**0.9654**
cnn_9	0.9552	**0.9612**	0.9722	**0.9809**	0.9378	**0.9413**	0.9547	**0.9607**
cnn_10	0.9547	**0.9562**	0.9739	**0.9742**	0.9351	**0.9378**	0.9541	**0.9557**
cnn_11	0.9688	**0.9759**	0.9842	**0.9909**	0.9533	**0.9609**	0.9685	**0.9757**

Tabela 6. Embedding FastText o długości 300, zbiór B

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	0.9297	**0.9303**	0.9294	**0.9299**	0.9310	**0.9318**	0.9302	**0.9309**
dense_2	0.9286	**0.9322**	0.9221	**0.9341**	**0.9373**	0.9310	0.9296	**0.9326**
dense_3	**0.9297**	0.9277	**0.9346**	0.9132	0.9250	**0.9463**	**0.9298**	0.9294
dense_4	0.9293	**0.9310**	0.9236	**0.9281**	**0.9371**	0.9353	0.9303	**0.9317**
lstm_1	0.9858	**0.9918**	0.9948	**0.9997**	0.9769	**0.9839**	0.9858	**0.9917**
lstm_2	**0.9918**	0.9909	**0.9994**	0.9972	0.9842	**0.9847**	**0.9917**	0.9909
lstm_3	**0.9920**	0.9904	**0.9992**	0.9967	**0.9849**	0.9842	**0.9920**	0.9904
lstm_4	0.9911	**0.9911**	0.9979	**0.9994**	**0.9844**	0.9829	0.9911	**0.9911**
lstm_5	**0.9916**	0.9911	**0.9987**	0.9969	0.9847	**0.9854**	**0.9916**	0.9911
lstm_6	0.9905	**0.9909**	0.9964	**0.9984**	**0.9847**	0.9834	0.9905	**0.9909**
lstm_7	**0.9921**	0.9900	0.9984	**1**	**0.9859**	0.9802	**0.9921**	0.9900
lstm_8	0.9920	**0.9921**	**1**	0.9992	0.9842	**0.9852**	0.9920	**0.9921**
lstm_9	**0.9925**	0.9914	**1**	0.9969	0.9852	**0.9859**	**0.9925**	0.9914
lstm_10	**0.9923**	0.9919	0.9989	**0.9997**	**0.9857**	0.9842	**0.9923**	0.9919
lstm_11	**0.9924**	0.9923	0.9994	**1**	**0.9854**	0.9847	**0.9924**	0.9922
lstm_12	**0.9921**	0.9919	0.9987	**0.9994**	**0.9857**	0.9844	**0.9921**	0.9919
lstm_13	0.9915	**0.9921**	0.9982	**0.9997**	**0.9849**	0.9847	0.9915	**0.9921**
lstm_14	0.9915	**0.9924**	0.9967	**0.9982**	0.9864	**0.9867**	0.9915	**0.9924**
lstm_15	0.9925	**0.9928**	0.9992	**0.9994**	0.9859	**0.9862**	0.9925	**0.9928**
lstm_16	0.9921	**0.9921**	0.9989	**0.9994**	**0.9854**	0.9849	0.9921	**0.9921**
cnn_1	0.9901	**0.9901**	**0.9977**	0.9964	0.9827	**0.9839**	0.9901	**0.9901**
cnn_2	0.9902	**0.9914**	0.9979	**0.9992**	0.9827	**0.9837**	0.9902	**0.9914**
cnn_3	**0.9875**	0.9866	**0.9964**	0.9956	**0.9787**	0.9776	**0.9874**	0.9865
cnn_4	0.9866	**0.9877**	**0.9959**	0.9951	0.9774	**0.9804**	0.9865	**0.9877**
cnn_5	**0.9896**	0.9889	**0.9977**	0.9974	**0.9817**	0.9804	**0.9896**	0.9888
cnn_6	0.9929	**0.9929**	1	**1**	0.9859	**0.9859**	0.9929	**0.9929**
cnn_7	0.9926	**0.9926**	0.9997	**0.9997**	0.9857	**0.9857**	0.9926	**0.9926**
cnn_8	0.9910	**0.9911**	**0.9992**	0.9979	0.9829	**0.9844**	0.9910	**0.9911**
cnn_9	0.9897	**0.9900**	**0.9984**	0.9982	0.9812	**0.9819**	0.9897	**0.9900**
cnn_10	**0.9905**	0.9886	**0.9979**	0.9961	**0.9832**	0.9812	**0.9905**	0.9886
cnn_11	**0.9918**	0.9916	**0.9992**	0.9989	0.9844	**0.9844**	**0.9917**	0.9916

Tabela 7. Embedding ELMo o długości 1024, zbiór B

	metryki							
X	accuracy		precision		recall		f1	
sieć	N	P	N	P	N	P	N	P
dense_1	0.9298	**0.9304**	0.9284	**0.9298**	0.93114	**0.93198**	0.93045	**0.93108**
dense_2	0.9294	**0.9323**	0.9220	**0.9337**	**0.93759**	0.93179	0.93017	**0.93298**
dense_3	**0.9301**	0.9282	**0.9340**	0.9126	0.9255	**0.94666**	**0.92994**	0.92959
dense_4	0.9297	**0.9319**	0.9234	**0.9273**	**0.93794**	0.93532	0.9307	**0.93254**
lstm_1	0.9860	**0.9922**	0.9940	**0.9994**	0.97705	**0.98439**	0.98642	**0.99252**
lstm_2	**0.9920**	0.9912	**0.9987**	0.9966	0.9844	**0.98543**	**0.992**	0.99098
lstm_3	**0.9927**	0.9911	**0.9986**	0.9959	**0.98559**	0.98435	**0.99286**	0.99065
lstm_4	**0.9920**	0.9914	0.9975	**0.9984**	**0.98524**	0.98304	**0.99189**	0.9914
lstm_5	**0.9919**	0.9916	**0.9978**	0.9967	0.98482	**0.98605**	**0.99208**	0.99192
lstm_6	**0.9913**	0.9912	0.9961	**0.9974**	**0.985**	0.98406	0.99134	**0.99184**
lstm_7	**0.9921**	0.9900	0.9981	**0.9998**	**0.98593**	0.98057	**0.99305**	0.99028
lstm_8	0.9925	**0.9927**	0.9990	**0.9991**	0.98485	**0.98581**	0.99246	**0.9931**
lstm_9	**0.9934**	0.9921	**0.9990**	0.9964	0.98605	**0.98606**	**0.99289**	0.99152
lstm_10	**0.9929**	0.9926	0.9984	**0.9990**	**0.9865**	0.98472	0.99235	**0.99236**
lstm_11	**0.9928**	0.9924	0.9985	**0.9990**	**0.98632**	0.98498	0.99256	**0.99282**
lstm_12	**0.9928**	0.9927	0.9977	**0.9986**	**0.98639**	0.98498	**0.99249**	0.99244
lstm_13	0.9918	**0.9928**	0.9972	**0.9996**	**0.98567**	0.98498	0.99171	**0.993**
lstm_14	0.9923	**0.9929**	0.9958	**0.9981**	0.98688	**0.98766**	0.99215	**0.99283**
lstm_15	0.9927	**0.9937**	0.9989	**0.9992**	**0.98662**	0.98655	**0.99304**	0.99301
lstm_16	0.9923	**0.9930**	0.9985	**0.9986**	**0.98604**	0.98511	**0.9928**	0.99251
cnn_1	0.9904	**0.9906**	**0.9968**	0.9955	0.98328	**0.98399**	0.99016	**0.99081**
cnn_2	0.9909	**0.9916**	0.9976	**0.9982**	0.98331	**0.9839**	0.99052	**0.9923**
cnn_3	**0.9875**	0.9872	**0.9957**	0.9946	**0.97881**	0.97797	**0.98777**	0.98652
cnn_4	0.9872	**0.9885**	**0.9952**	0.9950	0.97744	**0.98136**	0.98727	**0.98791**
cnn_5	**0.9905**	0.9890	0.9970	**0.9971**	**0.9823**	0.98082	**0.9906**	0.98882
cnn_6	0.9929	**0.9934**	0.9994	**0.9996**	**0.98672**	0.98624	**0.99376**	0.99332
cnn_7	0.9927	**0.9931**	**0.9992**	0.9988	**0.98617**	0.98586	0.99301	**0.99305**
cnn_8	0.9916	**0.9916**	**0.9984**	0.997	0.98312	**0.98467**	0.99141	**0.99189**
cnn_9	0.9899	**0.9905**	**0.9977**	0.9976	0.98182	**0.9826**	0.99001	**0.9904**
cnn_10	**0.9909**	0.9888	**0.9971**	0.9957	**0.98334**	0.98162	**0.99067**	0.98889
cnn_11	0.9924	**0.9925**	0.9983	**0.9986**	**0.98536**	0.98486	0.99175	**0.99182**

7.2.3. Wyniki dla eksperymentu 2

Tabela 8. Embedding Glove o długości 25, zbiór B jako zbiór treningowy, zbiór A jako zbiór testowy

	metryki			
sieć	accuracy	precisionaa	recall	f1
dense_1	0.559	0.5359	0.8785	0.6657
dense_2	0.554	0.5328	0.877	0.6628
dense_3	0.5545	0.5333	0.8715	0.6617
dense_4	0.5572	0.5352	0.8685	0.6623
lstm_1	0.5307	0.5165	0.9585	0.6713
lstm_2	0.52	0.5106	0.9575	0.6660
lstm_3	0.5405	0.5227	0.93	0.6693
lstm_4	0.5372	0.5213	0.911	0.6631
lstm_5	0.5362	0.5201	0.9375	0.6690
lstm_6	0.548	0.5272	0.929	0.6727
lstm_7	0.5362	0.5197	0.954	0.6728
lstm_8	0.527	0.5144	0.9585	0.6695
lstm_9	0.521	0.5111	0.9615	0.6674
lstm_10	0.5257	0.5137	0.959	0.6691
lstm_11	0.535	0.5192	0.945	0.6702
lstm_12	0.5317	0.5175	0.9385	0.6671
lstm_13	0.5297	0.5161	0.9495	0.6687
lstm_14	0.5195	0.5103	0.9655	0.6677
lstm_15	0.5335	0.5184	0.94	0.6683
lstm_16	0.5325	0.5176	0.9515	0.6705
cnn_1	0.5407	0.5243	0.877	0.6563
cnn_2	0.5372	0.5218	0.8885	0.6575
cnn_3	0.5392	0.5233	0.878	0.6558
cnn_4	0.538	0.5222	0.8905	0.6584
cnn_5	0.541	0.5245	0.875	0.6559
cnn_6	0.519	0.5099	0.9755	0.6697
cnn_7	0.5145	0.5074	0.9845	0.6697
cnn_8	0.5407	0.5240	0.889	0.6593
cnn_9	0.5375	0.5224	0.872	0.6534
cnn_10	0.533	0.5200	0.8575	0.6474
cnn_11	0.5412	0.5241	0.8945	0.6610

Tabela 9. Embedding FastText o długości 300, zbiór B jako zbiór treningowy, zbiór A jako zbiór testowy

	metryki			
sieć	accuracy	precisionaa	recall	f1
dense_1	0.5407	0.5228	0.9305	0.6695
dense_2	0.5412	0.5239	0.9025	0.6629
dense_3	0.5382	0.5218	0.914	0.6643
dense_4	0.541	0.5234	0.9145	0.6658
lstm_1	0.5095	0.5049	0.9765	0.6656
lstm_2	0.515	0.5078	0.9745	0.6676
lstm_3	0.5272	0.5144	0.969	0.6720
lstm_4	0.514	0.5072	0.9855	0.6697
lstm_5	0.5152	0.5079	0.97	0.6667
lstm_6	0.5152	0.5079	0.9775	0.6684
lstm_7	0.5187	0.5098	0.971	0.6686
lstm_8	0.5082	0.5042	0.976	0.6649
lstm_9	0.5065	0.5033	0.981	0.6653
lstm_10	0.509	0.5046	0.984	0.6671
lstm_11	0.504	0.5020	0.9885	0.6658
lstm_12	0.5032	0.5016	0.992	0.6663
lstm_13	0.5107	0.5055	0.9825	0.6675
lstm_14	0.505	0.5025	0.9915	0.6670
lstm_15	0.5065	0.5033	0.99	0.6673
lstm_16	0.5035	0.5017	0.9935	0.6667
cnn_1	0.5275	0.5146	0.963	0.6708
cnn_2	0.5257	0.5137	0.964	0.6702
cnn_3	0.5362	0.5198	0.9515	0.6723
cnn_4	0.5395	0.5213	0.9655	0.6770
cnn_5	0.5345	0.5185	0.9625	0.6740
cnn_6	0.5095	0.5048	0.989	0.6684
cnn_7	0.5032	0.5016	0.993	0.6665
cnn_8	0.5252	0.5134	0.962	0.6695
cnn_9	0.533	0.5179	0.954	0.6713
cnn_10	0.5345	0.5186	0.959	0.6732
cnn_11	0.5162	0.5084	0.9825	0.6700

Tabela 10. Embedding ELMo o długości 1024, zbiór B jako zbiór treningowy, zbiór A jako zbiór testowy

	metryki			
sieć	accuracy	precisionaa	recall	f1
dense_1	0.5328	0.5194	0.9244	0.6687
dense_2	0.5374	0.5212	0.9025	0.6626
dense_3	0.5345	0.5203	0.9049	0.6637
dense_4	0.536	0.5210	0.9088	0.6653
lstm_1	0.5002	0.4990	0.9707	0.6650
lstm_2	0.512	0.5043	0.9682	0.6672
lstm_3	0.5200	0.5076	0.9621	0.6715
lstm_4	0.512	0.5021	0.9826	0.6693
lstm_5	0.5108	0.4988	0.965	0.6661
lstm_6	0.5108	0.5074	0.9767	0.6677
lstm_7	0.5166	0.5090	0.9676	0.6682
lstm_8	0.5008	0.4998	0.9696	0.6645
lstm_9	0.506	0.4954	0.9768	0.6647
lstm_10	0.5014	0.4995	0.9803	0.6663
lstm_11	0.4986	0.4968	0.9863	0.6655
lstm_12	0.5001	0.5014	0.9886	0.6659
lstm_13	0.5097	0.5031	0.9776	0.6673
lstm_14	0.4992	0.4983	0.9907	0.6664
lstm_15	0.5013	0.4946	0.9835	0.6671
lstm_16	0.4956	0.4929	0.9887	0.6659
cnn_1	0.5212	0.5092	0.9575	0.6704
cnn_2	0.5190	0.5104	0.9626	0.6696
cnn_3	0.5265	0.5196	0.9428	0.6715
cnn_4	0.5351	0.5193	0.9618	0.6768
cnn_5	0.5263	0.5145	0.9574	0.6738
cnn_6	0.5029	0.4968	0.9862	0.6676
cnn_7	0.4942	0.4965	0.988	0.6665
cnn_8	0.5184	0.5044	0.9592	0.6693
cnn_9	0.5276	0.5124	0.9488	0.6710
cnn_10	0.5288	0.5112	0.953	0.6727
cnn_11	0.5152	0.5013	0.975	0.6695

7.2.4. Wyniki dla eksperymentu 3

Tabela 11. Embedding Glove o długości 25, zbiór B, ograniczony do równolicznych klas po 2000 rekordów

	metryki			
sieć	accuracy	precision	recall	f1
dense_1	0.8275	0.7985	0.8492	0.8230
dense_2	0.8275	0.7970	0.8518	0.8235
dense_3	0.8175	0.7944	0.8280	0.8108
dense_4	0.8325	0.7890	0.8809	0.8325
lstm_1	0.8937	0.8727	0.9074	0.8897
lstm_2	0.9337	0.9452	0.9126	0.9286
lstm_3	0.9412	0.9460	0.9285	0.9372
lstm_4	0.9375	0.9852	0.8809	0.9301
lstm_5	0.9437	0.9561	0.9232	0.9394
lstm_6	0.9287	0.9302	0.9179	0.9241
lstm_7	0.8975	0.9	0.8809	0.8903
lstm_8	0.9475	0.9516	0.9365	0.944
lstm_9	0.9625	0.9578	0.9629	0.9604
lstm_10	0.9675	0.9607	0.9708	0.9657
lstm_11	0.9525	0.9336	0.9682	0.9506
lstm_12	0.95	0.9642	0.9285	0.9460
lstm_13	0.9662	0.9705	0.9576	0.9640
lstm_14	0.9737	0.9709	0.9735	0.9722
lstm_15	0.95	0.9424	0.9523	0.9473
lstm_16	0.9387	0.9340	0.9365	0.9352
cnn_1	0.8725	0.8770	0.8492	0.8629
cnn_2	0.8737	0.8597	0.8756	0.8676
cnn_3	0.8637	0.8493	0.8650	0.8571
cnn_4	0.8937	0.8948	0.8783	0.8865
cnn_5	0.9275	0.9301	0.9153	0.9226
cnn_6	0.965	0.9704	0.9550	0.9626
cnn_7	0.9862	0.9893	0.9814	0.9853
cnn_8	0.8875	0.8730	0.8915	0.8821
cnn_9	0.8825	0.8776	0.8730	0.8753
cnn_10	0.8737	0.8488	0.8915	0.8696
cnn_11	0.8887	0.8714	0.8968	0.8839

Tabela 12. Embedding FastText o długości 300, zbiór B, ograniczony do równolicznych klas po 2000 rekordów

	metryki			
sieć	accuracy	precision	recall	f1
dense_1	0.9062	0.8796	0.9285	0.9034
dense_2	0.8975	0.8609	0.9338	0.8959
dense_3	0.8975	0.8609	0.9338	0.8959
dense_4	0.9162	0.8877	0.9417	0.9139
lstm_1	0.8737	0.8524	0.8862	0.8690
lstm_2	0.9687	0.9916	0.9417	0.9660
lstm_3	0.965	0.9781	0.9470	0.9623
lstm_4	0.9737	0.9890	0.9550	0.9717
lstm_5	0.9637	0.9754	0.9470	0.9610
lstm_6	0.9475	0.9468	0.9417	0.9442
lstm_7	0.9625	0.9702	0.9497	0.9598
lstm_8	0.9775	0.9838	0.9682	0.976
lstm_9	0.9812	0.9865	0.9735	0.9800
lstm_10	0.9862	0.9972	0.9735	0.9852
lstm_11	0.9787	0.9865	0.9682	0.9773
lstm_12	0.9712	0.9889	0.9497	0.9689
lstm_13	0.9887	0.9973	0.9788	0.9879
lstm_14	0.9862	0.9867	0.9841	0.9854
lstm_15	0.9862	0.9946	0.9761	0.9853
lstm_16	0.98	0.9813	0.9761	0.9787
cnn_1	0.9612	0.9676	0.9497	0.9586
cnn_2	0.9612	0.9651	0.9523	0.9587
cnn_3	0.9437	0.9393	0.9417	0.9405
cnn_4	0.9425	0.9300	0.9497	0.9397
cnn_5	0.9537	0.9595	0.9417	0.9506
cnn_6	0.99	0.9973	0.9814	0.9893
cnn_7	0.99	0.9973	0.9814	0.9893
cnn_8	0.9575	0.9623	0.9470	0.9546
cnn_9	0.9462	0.9419	0.9444	0.9431
cnn_10	0.9437	0.944	0.9365	0.9402
cnn_11	0.9587	0.9700	0.9417	0.9557

Tabela 13. Embedding ELMo o długości 1024, zbiór B, ograniczony do równolicznych klas po 2000 rekordów

	metryki			
sieć	accuracy	precision	recall	f1
dense_1	0.9212	0.9133	0.9206	0.9169
dense_2	0.9225	0.9136	0.9232	0.9184
dense_3	0.8912	0.8759	0.8968	0.8862
dense_4	0.9112	0.9050	0.9074	0.9062
lstm_1	0.9812	0.9814	0.9788	0.9801
lstm_2	0.9812	0.9865	0.9735	0.9800
lstm_3	0.9737	0.9685	0.9761	0.9723
lstm_4	0.955	0.9476	0.9576	0.9526
lstm_5	0.9562	0.9454	0.9629	0.9541
lstm_6	0.975	0.9637	0.9841	0.9738
lstm_7	0.97	0.9682	0.9682	0.9682
lstm_8	0.9862	0.9841	0.9867	0.9854
lstm_9	0.9887	0.9946	0.9814	0.9880
lstm_10	0.9862	0.9919	0.9788	0.9853
lstm_11	0.9912	0.9973	0.9841	0.9906
lstm_12	0.985	0.9972	0.9708	0.9839
lstm_13	0.9837	0.9866	0.9788	0.9827
lstm_14	0.9875	0.9919	0.9814	0.9867
lstm_15	0.9687	0.9632	0.9708	0.9670
lstm_16	0.9837	0.9790	0.9867	0.9828
cnn_1	0.96	0.9726	0.9417	0.9569
cnn_2	0.9675	0.9680	0.9629	0.9655
cnn_3	0.9287	0.9190	0.9312	0.9250
cnn_4	0.9187	0.9195	0.9074	0.9134
cnn_5	0.9425	0.9368	0.9417	0.9393
cnn_6	0.9925	1	0.9841	0.992
cnn_7	0.99	1	0.9788	0.9893
cnn_8	0.9437	0.9370	0.9444	0.9407
cnn_9	0.9512	0.9402	0.9576	0.9488
cnn_10	0.9362	0.9430	0.9206	0.9317
cnn_11	0.985	0.9945	0.9735	0.9839

8. Szczegółowy opis wykorzystanych architektur sieci

W tym rozdziale przedstawiona jest dokładna lista wykorzystanych w pracy architektur sieci, ich lista znajduje się w tabeli 15. Nastomiast w tableli 14 został umieszczony opis skrótów zastosowanych do opisu poszczególnych wartw.

Tabela 14. Opis oznaczeń wykorzsytanych w tabeli 15

Oznaczenie	Opis oznaczeń
Conv(n,k)	wartwa konwolucyjna, gdzie **n** - liczba zasosowanych filtrów, **k** - szerokość zastosowanego filtru
Flatten	warstwa powodująca spłaszczenie sieci do postaci jednowymiarowej
MaxPooling(f)	wartwa max poolingu o oknie wielkości **f** oraz skoku przesunięcia także wynoszacego **f**
Dense(n)	wartwa gęsta, o liczbie nueronów wynoszącej **n**
LSTM(n)	wartwa LSTM, liczbie nueronów wynoszącej **n**
Bi-LSTM(n)	wartwa Bi-LSTM, w której każda warstwa LSTM posiada **n** neuronów
Dropout(d)	parametr określający jak duża część połączeń zostanie usunięta między kolejnymi wartwami, wartość paramtru **d** określa jaka część losowych połączeń będzie usunięta w trakcie jednego kroku uczenia, zawiera się w przedziale od 0 do 1

Tabela 15. Architektury sieci wykorzystane w ramach pracy

Nazwa sieci	Sieć
cnn_1	Conv(50,3) Flatten Dense(50) Dense(1)
cnn_2	Conv(300,3) Flatten Dense(300) Dense(1)
cnn_3	Conv(300,15) Flatten Dense(300) Dense(1)
cnn_4	Conv(300,15) MaxPooling(2) Flatten Dense(300)

	Dense(1)
cnn_5	Conv(300,15) MaxPooling(8) Flatten Dense(300) Dense(1)
cnn_6	Conv(300,3) MaxPooling(2) Conv(300,3) MaxPooling(2) Flatten Dense(300) Dense(1)
cnn_7	Conv(300,3) MaxPooling(2) Conv(300,3) MaxPooling(2) Conv(300,3) MaxPooling(2) Flatten Dense(300) Dense(1)
cnn_8	Conv(300,5) Flatten Dense(300) Dense(1)
cnn_9	Conv(300,7) Flatten Dense(300) Dense(1)
cnn_10	Conv(300,9) Flatten Dense(300) Dense(1)
cnn_11	Conv(500,3) Flatten Dense(500) Dense(1)

$dense\_1$	Dense(30) Flatten Dense(1)
$dense\_2$	Dense(30) Dense(30) Flatten Dense(1)
$dense\_3$	Dense(300) Dense(300) Flatten Dense(1)
$dense\_4$	Dense(300) Dropout(0.2) Dense(300) Dropout(0.2) Flatten Dense(1)
$lstm\_1$	LSTM(50) Dense(5) Dense(1)
$lstm\_2$	LSTM(50) Dense(50) Dense(1)
$lstm\_3$	LSTM(30) Dense(30) Dense(1)
$lstm\_4$	LSTM(30) Dense(1)
$lstm\_5$	LSTM(30) Dense(1)
$lstm\_6$	Bi-LSTM(50)) Dense(1)
$lstm\_7$	Bi-LSTM(25)) Dense(1)
$lstm\_8$	LSTM(300) LSTM(300) Dense(1)
$lstm\_9$	LSTM(300)

	LSTM(300) Dense(300) Dense(1)
$lstm\_10$	LSTM(300) Dropout(0.2) LSTM(300) Dropout(0.2) Dense(300) Dense(1)
$lstm\_11$	LSTM(300) Dropout(0.2) LSTM(300) Dropout(0.2) LSTM(300) Dropout(0.2) Dense(300) Dense(1)
$lstm\_12$	Bi-LSTM(300) Dropout(0.2) Bi-LSTM(300) Dropout(0.2) Dense(300) Dense(1)
$lstm\_13$	Dropout(0.2) Bi-LSTM(300) Dropout(0.2) Bi-LSTM(300) Dropout(0.2) Dense(300) Dense(1)
$lstm\_14$	Bi-LSTM(500) Dropout(0.2) Dense(500) Dense(1)
$lstm\_15$	Bi-LSTM(300) Dropout(0.2) Bi-LSTM(300) Dropout(0.2)

	Dense(300) Dense(1)
$lstm\_16$	Bi-LSTM(300) Dropout(0.2) Bi-LSTM(300) Dropout(0.2) Bi-LSTM(300) Dropout(0.2) Dense(300) Dense(1)

9. Podsumowanie

W ramach pracy została dokonana kompleksowa analiza pozwalająca na zbudowanie modelu sieci neuronowej umożliwiającej rozwiązanie zadania klasyfikacji tekstów ironicznych. Zostały w niej zaprezentowane kolejne kroki, które należy wykonać by pozwolić sieci na interpretacje zdania. W pierwszej kolejności zostały omówione czynności niezbędne w ramach obróbki wstępnej danych. Następnie, przedstawiono sposoby przeniesienia zmiennych kategorycznych w postaci słów do przestrzeni wektorów liczbowych przy zachowaniu jak największej liczby informacji na temat podobieństwa poszczególnych słów. W końcowym etapie porównano kilka możliwych architektur sieci i ich skuteczność poprzez wykorzystanie różnych popularnych metryk. Ponadto zbadano wpływ cech morfosyntaktycznych na jakość klasyfikacji w problemie detekcji ironii. W oparciu o drobiazgowe wyniki nie dostrzeżono znaczącej poprawy jakości klasyfikacji dla żadnego z wykorzystanych zbiorów.

Niestety w ramach pracy nie wszystko poszło zgodnie z założeniami początkowymi. Nie udało się na przykład dokonać detekcji ironii w ramach języka polskiego. Było to spowodowane w dużej mierze brakiem łatwo dostępnych, bezpłatnych danych mogących posłużyć do analizy. W przypadku rozwiązania tego problemu, otwiera się bardzo ciekawy kierunek dalszego rozwoju tej pracy.

Innym elementem mogącym być dalszym kierunkiem rozwoju pracy jest opracowanie modelu pozwalającego na większą generalizację klasyfikacji dokonywanej przez model. W obecnej konfiguracji, sieć nie rodzi sobie dobrze z klasyfikacją wypowiedzi pochodzących z poza korpusu wykorzystanego w procesie treningu i testowania. Najprawdopodobniej wynika to ze zbyt małej różnorodności wypowiedzi w ramach zbioru danych. W celu rozwiązania tego problemu należałoby stworzyć korpus o bardziej różnorodnej budowie.

Bibliografia

[1] A. Radziszewski, **?**Metody znakowania morfosyntaktycznego i automatycznej płytkiej analizy składniowej języka polskiego**?**, prac. dokt., Wydział Informatyki i Zarządzania, Politechnika Wrocławska, 2012.

[2] http://blog.flocabulary.com/definitions-and-examples-of-irony-in-literature/.

[3] http://typesofirony.com/the-3-types-of-irony/.

[4] E. Riloff, A. Qadir, P. Surve, L. Silva, N. Gilbert i R. Huang, **?**Sarcasm as contrast between a positive sentiment and negative situation**?**, *Proceedings of EMNLP*, s. 704–714, sty. 2013.

[5] C. Baziotis, A. Nikolaos, P. Papalampidi, A. Kolovou, G. Paraskevopoulos, N. Ellinas i A. Potamianos, **?**NTUA-SLP at SemEval-2018 Task 3: Tracking Ironic Tweets using Ensembles of Word and Character Level Attentive RNNs**?**, sty. 2018, s. 613–621. DOI: 10.18653/v1/S18-1100.

[6] Y.-H. Huang, H.-H. Huang i H.-H. Chen, **?**Irony Detection with Attentive Recurrent Neural Networks**?**, kw. 2017, s. 534–540, ISBN: 978-3-319-56607-8. DOI: 10.1007/978-3-319-56608-5_45.

[7] S. Ilic, E. Marrese-Taylor, J. Balazs i Y. Matsuo, *Deep contextualized word representations for detecting sarcasm and irony*, wrz. 2018.

[8] J. Ling i R. Klinger, **?**An Empirical, Quantitative Analysis of the Differences Between Sarcasm and Irony**?**, t. 9989, maj 2016, s. 203–216, ISBN: 978-3-319-47601-8. DOI: 10.1007/978-3-319-47602-5_39.

[9] https://en.wikipedia.org/wiki/Word_embedding.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado i J. Dean, **?**Distributed Representations of Words and Phrases and their Compositionality**?**, *Advances in Neural Information Processing Systems*, t. 26, paź. 2013.

[11] J. Pennington, R. Socher i C. Manning, **?**Glove: Global Vectors for Word Representation**?**, t. 14, sty. 2014, s. 1532–1543. DOI: 10.3115/v1/D14-1162.

[12] P. Bojanowski, E. Grave, A. Joulin i T. Mikolov, **?**Enriching Word Vectors with Subword Information**?**, *Transactions of the Association for Computational Linguistics*, t. 5, lip. 2016. DOI: 10.1162/tacl_a_00051.

[13] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz i L. S. Zettlemoyer, **?**AllenNLP: A Deep Semantic Natural Language Processing Platform**?**, 2017. eprint: arXiv:1803.07640.

[14] https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk

Spis rysunków

Spis tabel