JanushPasswordAI

31 sierpnia 2019

Adam Chrzanowski
https://github.com/chradam

Aleksandra Marzec
https://github.com/AleksMarzec

Marcin Borzymowski
https://github.com/BMarcin

***Streszczenie*—JanushPasswordAI to sztuczna inteligencja, która wykorzystuje rekurencyjne sieci neuronowe *Reccurent Neural Netowrk* do tworzenia haseł podobnych do ludzkich, bazując na hasłach, które na co dzień wykorzystują polscy internauci. Projekt pokazuje jak ważne jest tworzenie silnych haseł i sposób zaangażowania sztucznej inteligencji w proces łamania haseł. Celem pośrednim projektu było utworzenie zależności pomiędzy literami ludzkich haseł, tak by znaleźć w nich jakieś schematy. Tym projektem chcemy zwiększyć świadomość internautów związaną z dbaniem o bezpieczeństwo w sieci.**

I. Wstęp

Hasła są najpopularniejszą metodą autoryzacji. Istnieje wiele powodów, dla których hasło połączone z nazwą użytkownika jest tak popularne. Przede wszystkim chodzi o łatwość implementacji oraz szerokie ich używanie przez większość społeczeństwa.

Ostatnimi czasy wycieki baz danych pokazały, że internauci nie przywiązują wystarczającej wagi do zabezpieczenia baz danych oraz używania mocnych haseł. Wiele baz nawet nie szyfrowało haseł, a inne korzystały z łatwej funkcji skrótu - MD5.

A. Odszyfrowywanie haseł w dzisiejszych czasach

W celu odszyfrowania haseł z wycieków baz danych najczęściej wykorzystuje się programy takie jak Hashcat[1]. Używają one kilku typów łamania haseł bazujących na ich zgadywaniu. Możemy dehaszować ciągi znaków bazując na słownikach, łączyć słowniki ze znakami/liczbami lub używać masek. Maski to swego rodzaju schematamy, według których należy tworzyć początkowe i/lub końcowe ciągi znaków dla każdego ze słów słownika. Takie metody dają bardzo dobre rezultaty, jednak my uważamy, że możemy zrobić to lepiej niż jakikolwiek program wykorzystujące standardowe metody.

B. Co oznacza słabe hasło?

Jak udało nam się do tej pory ustalić bardzo dużo osób tworzy hasła bezpośrednio z nimi związane. Najpopularniejszym jest ”123456”, a zaraz po tej kombinacji napotkamy ciągi będące nazwą domeny serwisu, w którym użytkownik się rejestrował. Dalej możemy wyróżnić kombinacje znaków ”qwerty” jak i imiona ukochanych osób połączone z jakąś datą. Ciekawostką (dość dla nas niespodziewaną) okazały się rejestracje samochodów.

[1]https://hashcat.net/hashcat/

C. Nasz cel

Jednym z celów naszego projektu jest znalezienie zależności jakie występują w hasłach stworzonych przez ludzi. Do wyznaczenia zależności między znakami haseł wykorzystamy model *CBOW - Continous Bag Of Words*. Zgodnie z początkowym planem projektu mieliśmy wykorzystać wyznaczone relacje do sieci GAN, jednak kierunek naszego projektu uległ zmianie. Sieci GAN zdecydowaliśmy się zastąpić sieciami, które idealnie nadają się do przetwarzania sekwencji - sieci rekurencyjnych. Uważamy, że nasze podejście jest lepsze niż samo zgadywanie haseł, gdyż sieć neuronowa jest w stanie ”zrozumieć” zależności między znakami w tworzonych przez ludzi ciągach znaków.

Model CBOW przyjmuje na wejściu litery słowa bez jednej, którą umieszczamy na wyjściu. W ten sposób przesuwamy nasze słowo przez model literka po literce i słowo po słowie. Sieć obliczy wagi reprezentujące zależności między znakami hasła w przestrzeni.

GAN jest siecią, która za wejście przyjmuje losowe ziarno, a na wyjściu daje nam ciąg znaków tworzący hasło. Zbudowane jest z dwóch sieci neuronowych - generatora (G) oraz klasyfikatora (D). Sieć G tworzy potencjalne hasło na podstawie otrzymanego ziarna. Sieć D natomiast ma za zadanie rozróżnić, czy hasło podane na wejściu jest hasłem, które można sklasyfikować jako ludzkie czy nie. Klasyfikator jest wykorzystywany tylko podczas uczenia, by nauczyć generator jak tworzyć odpowiednie ciągi znaków. Obie te sieci podczas nauki grają ze sobą w swego rodzaju grę. Im lepiej generator tworzy hasła tym bardziej klasyfikator rozpoznaje je (te pochodzące z generatora) jako błędne. Proces uczenia powinien trwać tak długo, aż klasyfikator zacznie uznawać hasła pochodzące od generatora jako te, pochodzące ze zbioru uczącego.

Sieci rekurencyjne są modelami, które wykorzystują sprzężenia zwrotne. Zmiana stanu jednego z neuronów będzie wpływać na inne neurony zgodnie z przepływem tensorów. RNN w odróżneniu od standardowych sieci liniowych charakteryzują się posiadaniem swego rodzaju pamięci i uzależniania od siebie kolejnych danych wejściowych.

Zbiór danych uczących jaki mamy pochodzi z wycieków baz danych z polskich serwisów internetowych. Głównie są to sklepy internetowe oraz fora. Większość z tych haseł musieliśmy deszyfrować własnoręcznie (Wielkie podzięko-

wania dla OVH! [2]). Do łamania haseł wykorzystaliśmy Hashcat'a z trybem 7 - wykorzystanie masek oraz haseł ze słownika. Maski były utworzone na podstawie jednego z najpopularniejszych zbiorów danych związanych z hasłami - RockYou[3]. Słownik jaki wykorzystaliśmy był całkiem duży (około 500 MB) i posiadał zarówno polskie jak i angielskie słowa. Ostatecznie udało nam się uzyskać 178 601 haseł.

Do sprawdzenia poprawności działania sieci wykorzystamy przygotowany przez nas zbiór haseł, który podzielimy w stosunku 95% - zbiór uczący i 5% - zbiór walidacyjny. Wygenerujemy próbki haseł, a następnie sprawdzimy ile z nich się powtórzyło w zbiorze walidacyjnym.

II. Powiązane prace

Nasz projekt jest inspirowany pracą PassGAN: A Deep Learning Approach for Password Guessing[4] opracowaną przez badaczy instytutu Stevens Institute of Technology w New Jersey oraz New York Institute of Technology. Przedstawiono tam wysoce skuteczną metodę odgadywania haseł opartą o sieć GAN. Przeprowadzone eksperymenty wykazują lepszą skuteczność zastosowania takiej metody w stosunku do najnowocześniejszych, opartych na regułach narzędzi takich jak HashCat i JTR. Należy jednak nadmienić, że dobre rezultaty zostały osiągnięte bez wiedzy o samych hasłach jak i ich strukturze. Bazując na dwóch zbiorach haseł pochodzących z wycieków RockYou oraz LinkedIn udało im się osiągność skuteczność dopasowań unikatowych haseł kolejno na poziomie 44% oraz 24%. Mimo niezbyt zachęcających wyników spora liczba wygenerowanych nieadekwatnych haseł przypominała te oryginalnie wytworzone przez człowieka, które potencjalnie mogły odpowiadać rzeczywistym hasłom nie branym pod uwagę w eksperymentach. Dodatkowo PassGAN w połączeniu z HashCat-em była w stanie poprawnie odgaadywać między 51%, a 73% więcej haseł niż sam HashCat. W założeniu PassGAN wytrenowana sieć autonomicznie określa cechy hasła oraz jego strukturę, by następnie na podstawie nabytej wiedzy wygenerować nowe próbki, które odpowiadają dystrybucji i do złudzenia przypominają rzeczywiste.

III. Opis metod

Działająca architektura sieci wykorzystuje jedynie rekurencyjne sieci neuronowe, a przygotowana sieć służąca do wyznaczania Character Embeddings nie została przez nas wykorzystana w dalszych etapach projektu. Z sieci GAN zrezygnowaliśmy po wielu testach zakończonych niepowodzeniami.

[2]https://www.ovh.pl/

[3]https://www.kaggle.com/wjburns/common-password-list-rockyoutxt

[4]https://arxiv.org/pdf/1709.00440.pdf

IV. Continous Bag Of Words

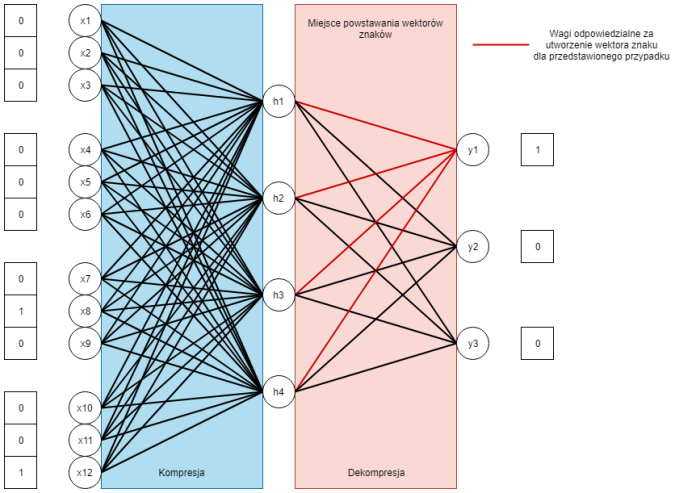
Celem użycia modelu CBOW jest wyznaczenie wektorów (punktów w przestrzeni) reprezentujących zależności dla poszczególnych znaków.

A. One hot vector

Przed podaniem hasła do modelu CBOW musimy przetłumaczyć je do postaci dzięki której sieć będzie w stanie łatwo wyznaczyć zależności znaków. W tym celu posłużymy się one hot vector-ami. Są to wektory składające się z *0* i *1* (zer i jedynek), które umożliwiają jednoznaczne identyfikowanie każdego znaku z podanego alfabetu. W takim wektorze znajduje się tylko jedna 1 (jedynka) i każdy one hot vector jest długości ilości unikalnych znaków alfabetu haseł. Ważną kwestią (którą omówimy w dalszej części) są puste wektory - niereprezentujące żadnego znaku. Ten specjalny typ wektora składa się z samych zer.

B. Budowa modelu

Sam model CBOW składa się z 3 warstw sieci: wejściowej, ukrytej i wyjściowej. Pomiędzy warstwą ukrytą, a warstwą wyjściową wyliczone zostaną wagi połączeń reprezentujące konkretne znaki. Ilość neuronów warstwy ukrytej odpowiada ilości wektorów jakie będą wyznaczone dla każdego znaku. Wagi odpowiedzialne za punkt w przestrzeni dla danego znaku możemy rozróżnić za pomocą neuronów wyjściowych - wagi będące przypisane dla neuronu pod którym występuje jedynka z one hot vector'a odpowiadają znakowi, który reprezentuje one hot vector. Poniższa grafika ilustruje zasadę działania. Na czerwono zaznaczono połączenia, których wagi odpowiadają wyznaczonym współrzędnym punktu w przestrzeni dla znaku, który reprezentuje one hot vector na wyjściu sieci.



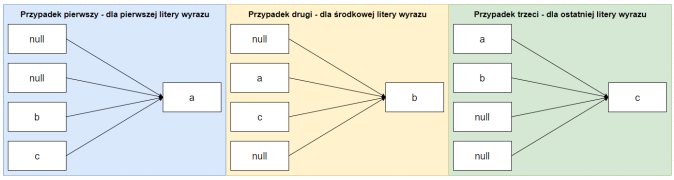
Rysunek 1. Podstawowy schemat modelu CBOW

C. Pętla działania CBOW

Dla każdego hasła musimy wyznaczyć wszystkie przestawienia znaków, tak by każdy z nich znalazł się na

wyjściu sieci. Zapomnijmy na chwilę o one hot vector'ach i przejdźmy na reprezentacje znakowe. Wyobraźmy sobie nasz model w postaci uproszczonej - jedynie wejścia oraz wyjście. Nasze wejścia dzielimy (tylko w wyobraźni) na pół. Górną połowę zajmą znaki poprzedzające znak wyjściowy, a dolną znaki występujące po znaku, który jest na wyjściu. W przypadkach znaków krańcowych umieszczonych na wyjściu, w zależności od znaku braki uzupełniamy pustymi one hot vector'ami.

Weźmy za przykład hasło 'abc'. Zgodnie z zasadą każda litera musi znaleźć się na wyjściu sieci, zatem w pierwszej iteracji literę 'a' dodajemy na wyjście naszego modelu. Górną część wejść (znaki poprzedzające) uzupełniamy pustymi wektorami, a dolne wektorami znaków 'b' oraz 'c'. W drugiej iteracji na wyjściu otrzymujemy literę 'b', w górną warstwę wejść musimy wpisać wektor zerowy oraz literę 'a', a w dolną warstwę literę 'c' oraz wektor zerowy. W ostatniej iteracji dla tego słowa na wyjście wpisujemy literę 'c', w górną warstwę wejść litery 'a' oraz 'b' i wreszcie w dolną dwa wektory zerowe. Powyższy przykład obrazuje następująca grafika. Oczywiście każdą z liter zastępuje one hot vector, a taką czynność przestawiania znaków należy powtórzyć dla każdego hasła.



Rysunek 2. Podstawowy schemat modelu CBOW

Powyższą logikę działania przenosimy na model CBOW. W trakcie uczenia możemy obserwować stosunkowo dużą wartość LOSS dla sieci. Wynika to bezpośrednio z naszych danych uczących. Dobrym przykładem będą ciągi znaków 'abc123' oraz 'abc133'. Sieć nie ma podstaw do wnioskowania, że w danej próbce hasła na wyjściu powinien pojawić się znak '2' lub '3'.

D. Optymizer oraz funkcja straty

Do uczenia sieci wybraliśmy optymizer Adam oraz *CrossEntropy* jako funkcję loss. We frameworku pyTorch funkcja *CrossEntropyLoss* zawiera w sobie również funkcję *SoftMax*. Dzięki funkcjiSoftMax Otrzymujemy dystrybucję prawdopodobieństwa wyliczanego na podstawie wartości wyjść sieci (wszystkie sumują się do 1). Następnie funkcja *CrossEntropy* wylicza różnice między predykcjami sieci, a poprawną wartością zbioru uczącego.

E. Opis eksperymentów

Udało nam się wyznaczyć 400 wymiarowe wektory *Character Embeddings* dla 100 000 haseł wejściowych. Hiperparametr *learning rate* ustawiliśmy na wartość 1^{-4}, liczbę epok na 10000 oraz *batchsize* na wartość 40000. Sieć trenowała się na karcie graficznej GTX 1060 3GB. Ograniczenia sprzętowe, a w szczególności brak optymalizacji algorytmu wpłynęły na długi czas uczenia oraz brak możliwości przeliczenia pełnego zbioru (w dalszej części pracy każda z sieci była optymalizowana tak, by jak najefektywniej wykorzystywać dostępne zasoby).

Zauważyliśmy, że sieć charakteryzowała się dość wysoką wartością funkcji *loss*, co naszym zdaniem uzasadnia przygotowany przez nas zbiór danych. Wiele z haseł miało podobny początek, ale odmienne końcówki np. 'kicia08' oraz 'kicia09'. Sieć nie miała najmniejszych podstaw, by wnioskować w jakikolwiek sposób która wartość przy predykcji danego przykładu jest właściwa - mowa tu o pojawianiu się na wyjściu znaku '8', a nie '9' przy haśle 'kicia09'. Takich przykładów było bardzo wiele, a wysoka wartość parametru *loss* będzie powszechnym zjawiskiem przy konstruowaniu kolejnych modeli opisanych w dalszej części pracy - w szczególności przy modelach rekurencyjnych.

Sieć CBOW nie charakteryzowała się większymi trudnościami związanymi z jej uczeniem, czy też ewaluacją. Spełniła swoje zadanie jak powinna. Sieć jednak mimo swej prostoty pozwoliła nam na zastosowanie wyznaczonych wektorów do sprawdzania podobieństw między literami (podobieństwo cosinusowe) jak i do sprawdzania poprawności podawanych do sieci słów. Największą ciekawość w naszym zespole wzbudziło wyznaczenie przez sieć grupy spółgłosek oraz samogłosek.

F. Ewaluacja sieci

Na etapie ewaluacji funkcja *Cross Entropy* nie jest wymagana. Do wyznaczenia predykcji wykorzystuje się jedynie funkcję *Soft Max*, a numer argumentu tensora wyjściowego pod którym znajduje się najwyższa wartość zwrócona przez funkcję odpowiada ID znaku będącego predykcją sieci.

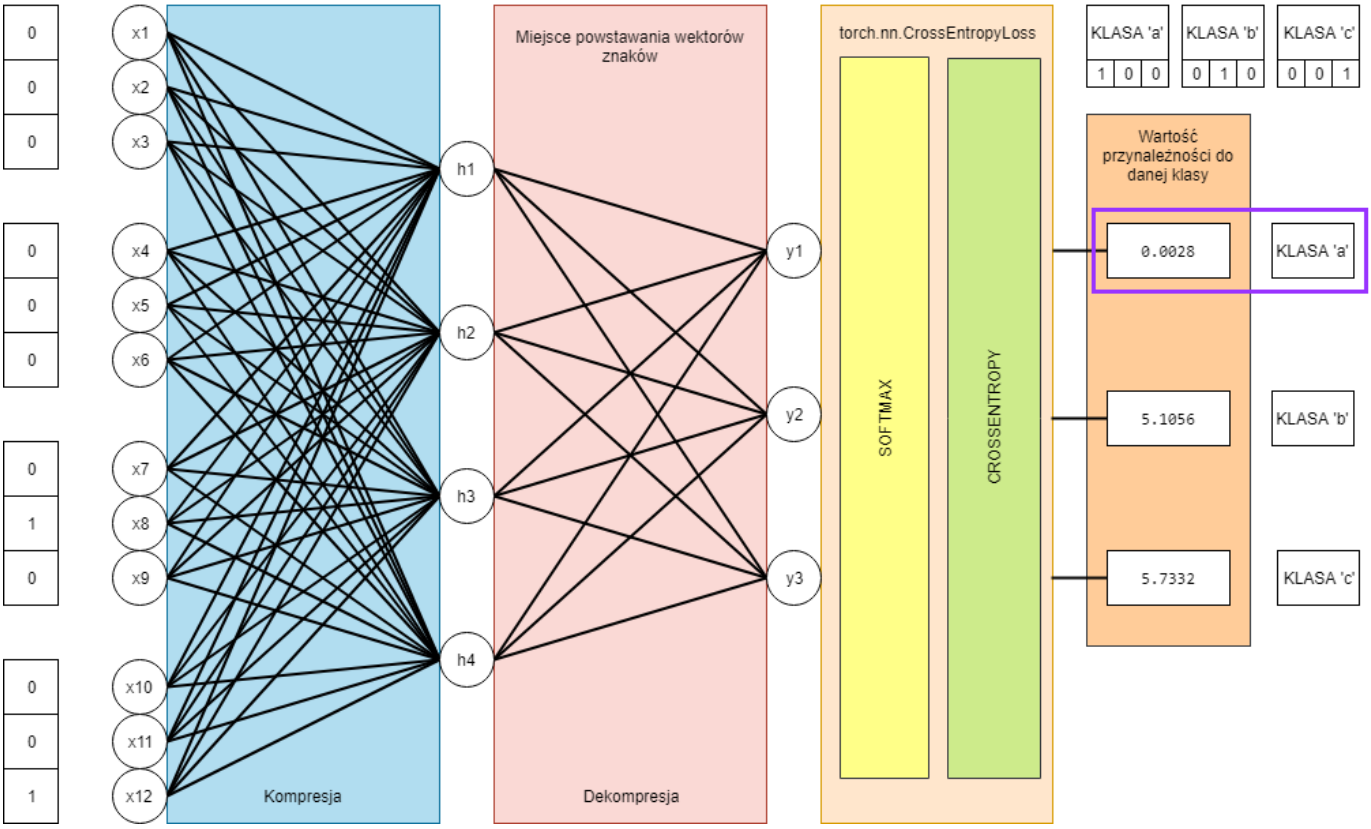
Na wejście sieci należy podać wszystkie znaki danego słowa z wyjątkiem jednego, którą model ma wskazać. Należy pamiętać o zastosowaniu *paddingów* zależnych od pozycji obliczanego znaku.

V. Generative Adversarial Network

W projekcie PassGAN wykorzystano sieci konwolucyjne, by sieci były w stanie generować sekwencyjne ciągi znaków.

A. Opis eksperymentów

W naszych eksperymentach przetestowaliśmy różne kombinacje architektur Generatora jak i Dyskryminatora poczynając od 1 warstwowych i 10 wejściowych sieci, a kończąc na 12 warstwowych modelach posiadających po kilkaset wejść. Mimo wyznaczania własnych funkcji *loss* żadna kombinacja architektur nie była w stanie generować dobrych próbek. Zauważyliśmy ponadto przypadłość sieci, które wykorzystywały funkcje aktywacju *ReLU* do wpadania w lokalne minima i mimo podawania zróżnicowanego ziarna na wejście generatora generowania tego samego ciągu co próbkę. Skutkowało to zmniejszaniem wartości



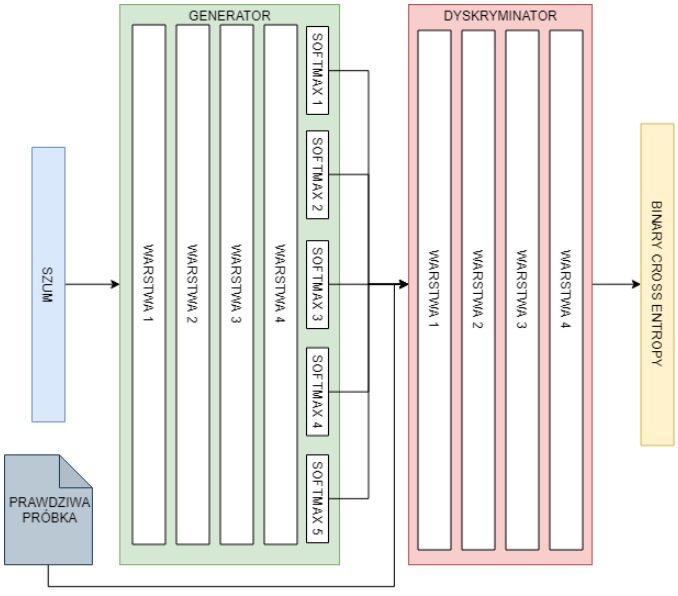
Rysunek 3. Schemat kompletnego modelu CBOW

loss dla Dyskryminatora i zwiększania tej wartości dla Generatora. Stosowanie *dropoutów* dodatkowo nie przyniosło żadnych skutków. Stwierdziliśmy, że pomóc w tym może zastosowanie własnej (dodatkowej) funkcji straty jak i zmniejszenie wartości *learning rate* dla Dyskryminatora. Jednak nic nie przyniosło większych skutków.

1) Sieci oparte o Character Embeddings: Na samym początku zbudowaliśmy architekturę, która każdy znak z próbki rzeczywistej tłumaczyła na wyliczone za pomocą modelu CBOW *Character Embeddings* i podawała je do Dyskryminatora. Generator posiadał liczbę wyjść wynoszącą *wielkość okna * wymiarowość wektorów znaków*. W ten sposób do wyniku otrzymanego z funkcji straty dodawaliśmy wynik z funkcji straty, która obliczała podobieństwo pomiędzy predykcją wektora znaku, a najbliższym wyznaczonym przez model CBOW.

2) Sieci oparte o ID znaków: Stwierdziliśmy, że korzystanie z wektorów znaków nie ma większego sensu, gdyż model Generatora będzie musiał wyznaczyć je samemu.

Idea posiadania unikalnego ID przez każdy ze znaków dawało ogromną oszczędność pamięci w porównaniu do poprzedniej koncepcji. Własna funkcja straty polegała na zliczaniu ilości pustych znaków w generowanych hasłach. Zauważyliśmy, że sieć wypełniała w 100% dostępne okno i dlatego postanowiliśmy wymusić na modelu generowanie krótszych ciągów znaków (im więcej pustych znaków sieć generowała tym wartość funkcji straty malała) z nadzieją na polepszenie tworzenia trafnych sekwencji.



Rysunek 4. Jedna z testowanych architektur GAN

B. Przejście z GAN na RNN

W trakcie tworzenia projektu mieliśmy styczność z tematem sieci rekurencyjnych, sieci *seq2seq*(sieci wykorzystywane m.in. do tłumaczeń) oraz modeli językowych, które podsunęły nam pomysł przejścia z liniowych sieci GAN na właśnie sieci rekurencyjne wykorzystywane przy zadaniach związanych z sekwencjami.

VI. Recursive Neural Network

W dzisiejszych czasach sieci RNN wykorzystuje się w wielu zadaniach związanych z NLP jak i modelowaniu np. zużycia prądu na podstawie serii pomiarów. Sieci tego typu można również wykorzystać do tworzenia systemów rekomendacyjnych opartych o sekwencje wyborów użytkownika np. serwisy muzyczne czy też wideo.

A. Inspiracja modelami językowymi

Modele językowe takie jak *BERT*[5] oparty o *Transformery*[6] (które w pewnym momencie chcieliśmy wykorzystać w naszym projekcie, jednak uznaliśmy, że mają one lepsze zastosowanie w zadaniach NLP niż modelowania haseł), *ELMO*[7] oparty o dwukierunkowe komórki *LSTM* czy też *ULMFiT*[8] oparty o komórki *AWD-LSTM*[9] stały się naszą inspiracją do stworzenia własnego modelu sekwencyjnego. Widzieliśmy w akcji *BERT'a* oraz *ULMFiT'a* jak generowały tekst, więc stwierdziliśmy, że pewnie w równie dobry sposób poradzą sobie z generowaniem haseł. Jednym z pomysłów jaki mieliśmy było wykorzystanie przeuczonego modelu *UMLFiT* do naszego projektu, jednak stwierdziliśmy, że byłby to przerost formy nad treścią, więc ostatecznie przygotowaliśmy własną architekturę. Dodatkowym problemem byłoby samo dostosowanie danych wejściowych do architektury modelu.

B. Komórka GRU

Wiedząc, że sieci rekurencyjne pomogą nam w zbudowaniu modelu, który będzie spełniać swoje zadanie natrafiliśmy na dwie możliwości - LSTM oraz GRU. Obie komórki miały za zadanie niwelować problem zanikających gradientów, czyli sytuację w której podczas wstecznej propagacji wraz z każdą warstwą sieci rekurencyjnej wartości gradientów malały tak, że pierwsza warstwa otrzymywała znikome wartości w trakcie uczenia. Wygasające gradienty spowalniają proces uczenia w znacznym stopniu. LSTM jak i GRU mają bardzo podobną charakterystykę działania i można je wręcz wymiennie stosować przy tego samego typu problemach. Badania pokazuja, że starsze komórki (LSTM) zachowują się lepiej przy modelach *seq2seq* na dużą skalę (np. tłumaczenia długich tekstów). Uznaliśmy, że komórki GRU spełnią lepiej swoje zadanie głównie ze względu na brak konieczności przyjmowania dodatkowego parametru jakim jest *cell state* (jak to jest w przypadku LSTM), co dawało mniejsze wykorzystanie pamięci kart graficznych.

GRU - *Gated Recurrent Unit* został wprowadzony w 2014 roku i charakteryzuje się posiadaniem dwóch bramek - *update* oraz *reset*. Bramki posiadają funkcje aktywacji - sigmoid oraz tangens hiperboliczny, które decydują o przepływie tensorów generując wartość wyjściową oraz kolejny stan macierzy ukrytej.

C. Budowa modelu

Zbudowany przez nas model jest modelem, który reprezentuje prawdopodobieństwo warunkowe występowania znaków pod warunkiem występowania znaków go poprzedzających. Z tego też powodu predykcją otrzymaną od modelu jest jedynie ostatni wektor zwracanej przez model macierzy. Wektor ten podczas uczenia przechodzi dalej przez funkcję *SoftMax* oraz *CrossEntropyLoss*.

Architektura modelu opiera się o dwukierunkowe komórki GRU. Dają one lepsze predykcje, gdyż przy predykcjach biorą one pod uwagę podawane wejście od lewej do prawej strony oraz od prawej do lewej. Do zadania generowania haseł ma to kluczowe znaczenie, gdyż końcówki haseł (najczęściej cyfry) mają kluczowe znaczenie - w dzisiejszych czasach mało kto będzie na końcu używać roku np. 1200, a bardziej lat w okolicy roku 2000.

Przed wejściem każdej komórki, która otrzymuje wektor danego znaku znajduje się warstwa *Embedding*. Za pomocą otrzymanego ID znaku przy użyciu m.in. metod normalizacji generuje wektor znaku zbliżony do *OneHot* wektorów. Pod indeksem ID znaku podaje najwyższą wartość spośród pozostałych.

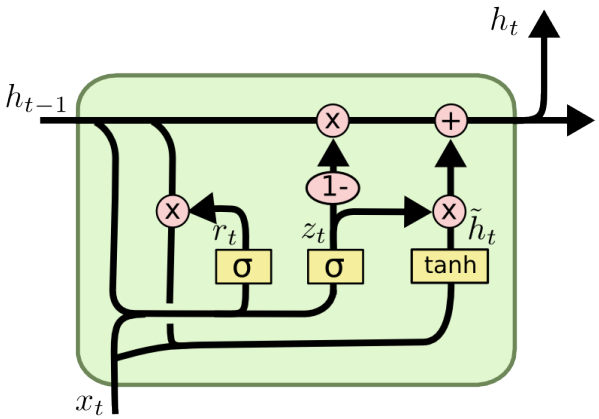
D. Dane wejściowe i wyjściowe

Dla każdego unikalnego znaku ze zbioru haseł należało wyznaczyć unikalne ID oraz do takiego słownika dodać element '<EMPTY>' oznaczającego pusty znak. Model nasz jest modelem sekwencyjnym, więc również i dane uczące musiały być w taki sposób podawane. Każde słowo było rozbijane na słowa składające się z sekwencji zawierających od początkowego znaku aż do każdego kolejnego znaku - dane wejściowe, a na wyjściu zastosowano przesunięcie o jeden w przód, a za ostatnim znakiem każdego hasła, jak i w dopełnieniach do wielkości okien wstawiono 'znaki <EMPTY>'. Przykładowo dla hasła 'kicia08' i wielkości okna 9 pierwsza sekwencja będzie mieć następującą postać (wejscie, wyjście): ([k, <EMPTY>, <EMPTY>, <EMPTY>, <EMPTY>, <EMPTY>, <EMPTY>], [i]) i tak aż do ostatniej sekwencji jaką jest ([k, i, c, i, a, 0, 8, <EMPTY>, <EMPTY>], [<EMPTY>]).

E. Dropout i funkcje aktywacji

Postanowiliśmy pozostawić warstwę *Dropout* ze względu na świetne wyniki stosowania tej funkcji w komórkach *AWD-LSTM*. Warstwa ta zgodnie z podanym prawdopodobieństwem według rozkładu normalnego zeruje losowe

[5]https://arxiv.org/pdf/1810.04805.pdf
[6]https://arxiv.org/abs/1706.03762
[7]https://arxiv.org/abs/1802.05365
[8]https://arxiv.org/abs/1801.06146
[9]Są to sieci, które w pełni oparte są o dropouty. Zastosowano je do zerowania wektorów wejściowych czy też wartości wektorów ukrytych przekazywanych między komórkami



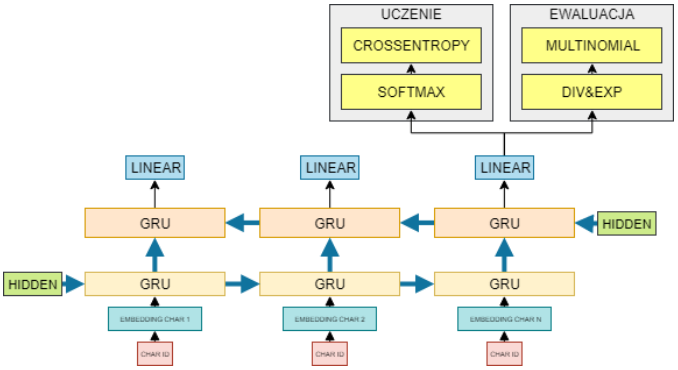
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Rysunek 5. Schemat komórki GRU



Rysunek 6. Schemat kompletnego modelu znakowego

liczby w tensorze wejściowym. Działanie to zapobiega przeuczeniu sieci wydłużając czas uczenia modelu przy zachowaniu lepszych predykcji (w naszym przypadku) sieci liniowej występującej na wyjściu komórki GRU.

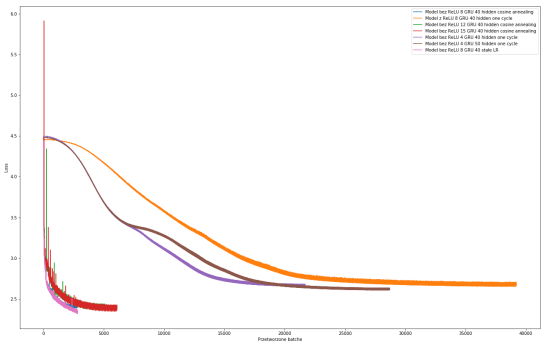
W wyniku przeprowadzania wielu testów zauważyliśmy, że stosowanie funkcji aktywacji *ReLU* powodowało częste powtarzanie się tych samych sekwencji w generowanych hasłach - chodzi o sekwencje w stylu: samogłoska, spółgłoska, samogłoska, spółgłoska głównie po pierwszym znaku hasła. Był to dobry powód do rezygnacji z tychże funkcji.

F. *OneCycle learning*

Stosowanie zmiennej wartości *learning rate* po raz pierwszy zauważyliśmy w kursie Fast.AI[10] przygotowanym przez Jeremego Howarda. Wysunął on ideę do zmieniania wartości *learning rate* optymizera w trakcie procesu uczenia, tak by na początku wpaść w lokalne minimum (poprzez niską wartość), następnie je przeskoczyć (stosując dużą wartość), by dalej znaleźć to minimum (ponownie stosując niską wartość). Zdecydowaliśmy się zaimplementować tą ideę i przetestować uczenie modelu (w sposób liniowy osiągając maksymalną wartość w połowie epok uczenia). Należy tutaj zwrócić uwagę na kwestię zachowywania momentów optymizera - tak dostosować kod, by zmienić jedynie parametr *learning rate*, a zachować obliczone wcześniej momenty. Zastosowanie tego typu metody spowodowało, że do osiągnięcia niskich wartości funkcji straty należało wykonać 700 epok, co trwało około 5 godzin.

G. *Cosine Annealing learning*[11]

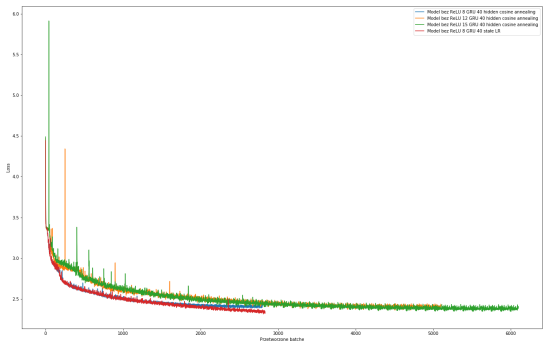
Zdecydowaliśmy się również przetestować zmianę wartości *learning rate* według funkcji *cosinus* zaczynając od wysokiej wartości i schodząc do ustawionej wartości minimalnej - w naszym przypadku było to 1^{-7}. Mimo osiągnięcia lepszego wyniku nie stosując zmiennej wartości *learning rate* zdecydowaliśmy się na ewaluacje modelu posiadającego 15 komórek GRU i 40-sto wymiarowej macierzy ukrytej. Zastosowanie tego typu metody spowodowało, że do osiągnięcia niskich wartości funkcji straty należało wykonać zaledwie 50 epoch, co trwało ok. 40 minut.



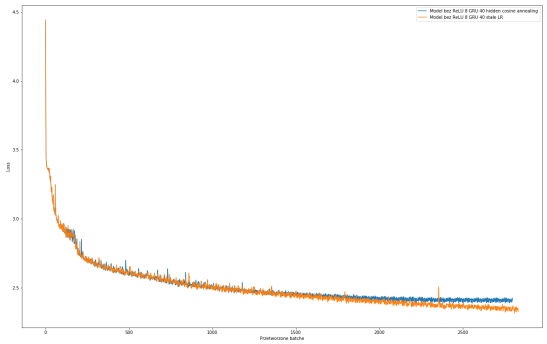
Rysunek 7. Porównanie minimalizacji funkcji straty różnych konfiguracji modeli

[10] https://course.fast.ai/videos/

[11] https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.CosineAnnealingLR



Rysunek 8. Porównanie minimalizacji funkcji straty różnych konfiguracji modeli (przybliżenie)



Rysunek 9. Porównanie minimalizacji funkcji straty modelu ze stałą wartością learning rate oraz z cosine annealing

H. Optymizer oraz funkcja straty

Najlepszym wyborem dla funkcji *loss* dla przygotowanej architektury była funkcja *CrossEntropyLoss*. Dzięki niej byliśmy w stanie obliczyć o ile w predykcjach różni się sieć od zakładanej wartości wyjścia.

Podczas testów wykorzystaliśmy dwa optymizery - *Adam* oraz *RMSProp*. Drugi z nich wypadał podczas uczenia lepiej, więc zdecydowaliśmy się przy nim pozostać.

Chcieliśmy nadmienić jeszcze, że na kilka dni przed zakończeniem tworzenia ostatnich modeli naszego projektu wypuszczono nowe metody znajdowania minimum takie jak: RAdam[12] oraz *Lookahead*[13], które mogłby polepszyć jakość generowanych haseł.

I. Opis eksperymentów

Uczenie modeli rekurencyjnych odbywało się na karcie GTX 1080 Ti 11GB. Wartość *batch size* była ustawiana w zależności od wielkości modelu jaki testowaliśmy. Niektóre modele powodowały ograniczanie *batcha* do wartości 10 000, a inne pozwoliły na wartość 50 000. Początkowo popełniliśmy błąd i trenowaliśmy modele bardziej pod klasyfikowanie niż generowanie ciągów znaków.

[12]https://arxiv.org/pdf/1908.03265.pdf
[13]https://arxiv.org/abs/1907.08610

Tablica I
Przygotowane modele

GRU	Hidden	F. aktyw.	Metoda *lr*	Epoki
8	40	ReLU	OC od 1e-2 do 1e-8	700
8	40	Brak	CA od 1e-3 do 1e-8	50
4	40	Brak	OC od 1e-2 do 1e-8	700
4	50	Brak	OC od 1e-2 do 1e-8	700
8	40	Brak	Stała - 1e-3	50
12	40	Brak	OC od 1e-2 do 1e-8	50
15	40	Brak	OC od 1e-2 do 1e-8	50

J. Porównanie różnych architektur

Nie byliśmy do końca pewni jakie parametry będą najlepsze dla naszego zadania. Przygotowaliśmy kofiguracje, które obrazuje tabela 1. Osiągnięte wartości funkcji straty obrazują umieszczone w dokumencie wykresy.

K. Ewaluacja modelu

Do ewaluacji zdecydowaliśmy się na wybranie modelu, który posiada 15 warstw dwukierunkowych komórek GRU, wielkość macierzy *hidden* wynoszącą 40 i był uczony przy użyciu metody *Cosine Annealing* przez 50 epok. Liczba parametrów modelu, która podlegała uczeniu wynosiła 455 992.

Podchodząc do tworzenia sieci rekurencyjnych wiele wątpliwości wzbudziła u nas możliwość generowania sekwencji, gdyż wiemy, że sieć nauczyła się rozkładu prawdopodobieństwa i dla zadanego ciągu będzie podawać zawsze te same - najbardziej prawdopodobne predykcje. Do rozwiązania tego problemu zainspirowało nas rozwiązanie stosowane przy modelach językowych.

Każdą z wartości otrzymanego wektora pochodzącego z ostatniej warstwy rekurencyjnej dzielimy przez parametr *temperature*. Im większa wartość tego parametru, tym generowane hasła będą bardziej się od siebie różnić i tym bardziej będą odbiegać od wyuczonych schematów. Przy modelach językowych zaleca się używanie wartości 0.75 tego parametru, a my przy naszej ewaluacji zdecydowaliśmy się na użycie 0.6 ze względu na ilość generowanych próbek. Następnie po podzieleniu wyznaczamy wartość wykładniczą. Uzyskane w ten sposób wartości przepuszczamy przez wielomianową dystrybucję prawdopodobieństwa, tak by uzyskać jedną wartość dla każdego kolejnego znaku hasła.

$x_i = multinomial(e^{x_i/temperature})$

Do obliczenia wyników eksperymentów wygenerowaliśmy 131 000 haseł (dla przypomnienia: zbiór walidacyjny posiada 8931 haseł). Ze względu na taką właśnie ilość parametr *temperature* ustawiliśmy na wartość 0.52, jednak przy jednostkowym generowaniu polecamy niższe wartości np. 0.45. By wygenerować hasło należy podać znak startowy ze słownika znaków (znaków, które znalazły się w zbiorze uczącym). W trakcie ewaluacji skorzystaliśmy z

losowego wybierania znaku startowego. By model samemu wybierał nawet znak początkowy haseł należałoby nauczyć go, że hasła rozpoczynają się od sekwencji np. '<START>' podobnie jak to ma zastosowanie w przypadku sekwencji '<END>'.

L. Otrzymane wyniki

Porównywanie wyników w takiej kwestii jak generowanie haseł jest dość problematyczne do określenia poprzez liczby. Masa haseł generowana przez model wygląda jak hasła podobne do zbioru uczącego czy też walidacyjnego. Problem pojawia się przy porównywaniu, gdyż model musi trafić idealnie w hasło znajdujące się w zbiorze walidacyjnym. Chodzi nam tu o przypadek, w którym model wygeneruje hasło 'haslo123', a w zbiorze walidacyjnym znajdowało się hasło 'haslo120'. Konstrukcja jest podobna i prawdopodobieństwo występowania takiego hasła jest w zasadzie identyczne. Mimo to udostępniamy liczbowe porównanie.

VII. Podziękowania

Przede wszystkim chcielibyśmy podziękować firmie theBlue.ai[14], która udostępniła nam nielimitowany dostęp do własnych zasobów obliczeniowych, by móc trenować wiele modeli jednocześnie, co w znaczący sposób przyspieszyło prace nad projektem i umożliwiło przygotowanie działającego modelu oraz pracownikom za wskazanie drogi jaką były sieci odwzorowujące sekwencje.

VIII. Podsumowanie

Nasz projekt pokazuje w bardzo dobry sposób, że można wykorzystać sieci neuronowe do łamania zabezpieczeń internautów. W trakcie projektu nauczyliśmy się bardzo wiele i przeszliśmy od aktywnego wykorzystywania prostych 3-warstwowych sieci liniowych aż do dużych rekurencyjnych sieci modelujących złożone sekwencje.

Uzyskany przez nas model stanowi idealne dopełnienie innych metod łamania haseł, a wygrywa z nimi o tyle, że posiada wyuczone reguły tworzenia haseł, których przeniesienie na programy typu hashcat jest niemożliwe.

Spędziliśmy wiele godzin poszukując odpowiednich rozwiązań i jeszcze więcej godzin na szkoleniu modelów. Od końca czerwca do końca sierpnia praktycznie każdej nocy szkolił się minimum jeden model.

W przyszłości warto by było przeuczyć model, by samemu wybierał znaki startowe haseł. Uważamy, że dałoby to lepszej jakości ciągi znaków.

Cały przygotowany przez nas kod znajduje się pod adresem https://github.com/BMarcin/JanushPasswordAI

IX. Na samym końcu chcieliśmy udokumentować tworzone hasła

Wszstkie hasła umieszczone poniżej związane z pokryciami zbiorów walidacyjnego lub uczącego są losowo wybrane. Najlepszy obraz efektywności modelu daje ostatnia podsekcja tej sekcji, której początkowe znaki zostały ograniczone do cyfr i znaków alfanumerycznych.

A. Poniższe hasła pochodzą ze zbioru pokyrwających się haseł wygenerowanych przez model z hasłami ze zbioru ***walidacyjnego***

sara13, lolek1234, neo24121, tomek1234, jacek1989, neo1233, neo2400, mala1984, 30091973, 240683, tomek16, barusia1, ola2005, 180383, misio123, ewa1975, Elunia1, irena1970, 13081988, marek1971, marta123, spinka1, 30011984, julia2006, marek1979, gosia1986, neo2411, ania1974, neo2407, 051279, 071081, mania1977, gosia2006, daga1234, neo2412, mario1975, ewakom1, ania1976, ela1976, worek1234, 291284, ania68, Kasia54, mariola1, 14071982, 28031974, asia1234, buba123, Putek123, 17081971, Kasia2006, 210661, neo2475, 12121981, marta22, inter1234, Ewa1970, ewa1949, jacolek, marus19, 220765, 03092000, 010390, tomek82, Marek1979, violka1, sonia2010, jaro1984, fina1234, jasio78, 15061979, 260579, jola123, 09011983, lolek123, 03101976, 09051985, ryfa1234, 820302, monia1985, emirek1, ewela1983, kasia1988, 08121986, 29091976, Anusia1, 09031980, amelcia, aga1234, 02031984, 22021958, 120271, tomek1992

B. Poniższe hasła pochodzą ze zbioru pokrywających się haseł wygenerowanych przez model z hasłami zbioru uczącego.

tina1234, viki2000, jacek1974, sara2002, inka1979, lolo1234, darek1976, Asia2007, 22081982, 12121964, sonia1975, ela2010, Kasia1986, daria1986, sonia1977, ania1234, 29011982, Gosia1977, kasia2009, pikus19, oskara1, Marek1971, kuba2006, 08071982, kasia1234, 23051983, czarka1, gosia1983, 260784, 18091962, 0000000, Radek1234, adamus1, 16051984, adam123, 08071977, wiola1978, golfik1, Martusia, ryba1987, Hania123, Marek1980, Ania1981, olika2000, krata666, kasia21, Ania1981, marbi19, kasia21, jasio2008, Romek123, jola1985, 081283, tomek1967, kasia74, aga1988, 720601, 20121977, dorek1979, 26071975, neo2410, neo1980, czesia1, ola1984, Inka1969, radek2007, kasia1973, marta1974, marta1981, 30031975, ewa1984, inka1983, inter123, kamarek, 123maria, 740402, 310880, kasia30, rokula1, ania1978, 31051970, 30041962, olek1972, viki1977, 210688, natanek, darek1973, Kasia123, neo2406, kasia1973, ania1979, abasia1, kotek1985, natala1, marta1978, 25051985, neo2427, kasia2006, tomek1985, gonia68, 06071973, Hania2000, darek83, pati1981, 11021983, 23031983, Kasia1984, kasia111, tomek1977, jola1975, marta26, 19831983, bart1978, basia1980, 08051984, neo2413, koniki1, 24031978, 050779, Tomek2006, 28121983, hania72, wiki2009, marta1980, 21011985, mika18, 090983, 231083, eko123, wiki2001, turas123, 20021970, Ania1983, ola2008, neo24001, gosia2009, ewelka1, neo24100, Kasia1986, 211057, ola1983, adam1234, basia1980, praca1234, marta13, jasia123, asia2012, 24011980, Inga2004, kris1110,

[14]https://theblue.ai/pl/

asia1972, tomek1985, kasia19, darek1967, ania1973, Dania2006, witek1976, neo1979, mania2001, ania1996, darek1977, tomek1983, Baran1982, 100284, ela1983, 07061978, nina2001, 300385, lena2006, radek1979, marta78, jola1234, jacek2009, 30041982, kasia1987, pati1234, mariak1, Marta1985, 01041982, 24121974, Kasia1983, 200471, darek1970, 27031972, 071178, Ania1986, ania2008, gama1969, marusia1, 23081975, art1974, marta1984, Ania123, darek1978, ania2005, kasia13, 210381, 030684, 04021979, Marta123, halina123, 230771, janek1980, ania123, 21041979, kajka2009, 20031992, gosia1989, Nora2007, kotek1985, marta1975, 250584, zaba1970, ola2008, ania1977, Ewunia1, 30031985, golas123, sonia1975, nata123, Hania123, janek1984, Ania1985, wiolka1, kasia1234, basia1976, 070875, 02021966, 07021970, Ela123, maria1979, kasia2005, 20051974, Kuba2003, julka1974, 31031969, Marek83, 28062003, ania997, 180671, Marta2004, 710305, 21051978, ania2008, olinka1, ania1987, Gosia111, ola2003, pola1234, dupa1000, barton1, daro1234, dupa2007, marta1983, 24051986, radek1988, 30081978, 20071988, Marta1987, Hania2005, miki1979, ania1975, ela12344, 060577, Marta2004, Gosia666, hania2003, gaja1977, radek1978, 17061983, asia75, asia1970, ela12344, kinia1984, tomanek, marta17, 270783, ania1234, gosia1973, viki1977, 270387, lodzia1, Ania1981, ania2009, 170886, jarek1961, legia13, 13091984, kasiak1, kasia1970, tomek1980, Radek123, 221078, 30061973, 220675, gosia2009, Tomek1984, 12121990, lolek1983, kuba2001, pati1975, lolek13, 14041977, 02101981, rafik13, 220676, ela12345, ola2007, sara2008, wiki2007, marta1978, 23021988, ania1979, Magda1969, jula2002, inka21, 050169, ak1982, wiki2004, 160683, gosia1983, 151286, karola1, adam123, indorek, marta100, 31081980, 22011983, ania1983, ola2006, 19121983, michas1, szansa1, 01081980, marek1982, wiki2008, 02041979, marta83, 24071979, Marta1984, neo242000, ola2010, ewa1978, 241081, Marta1980, danka13, 30071976, Marta01, ania1988, tomek21, marta77, basia1987, 300405, marta1986, ewa2006, 08021971, hania2003, darek1983, jacek1977, ania1981, 201183, bartas1, Ania1981, 4113274, ewka100, ania1975, Marek1962, marianka, irek72, tomek19, Kasia1974, nora2000, 21051982, 15061973, 01051974, Marta1984, 22081976, ewa1980, neo1974, 27111984, 25051985, karola1, tomek1985, sonia87, adam50, 22081982, 25061976, 1234kasia, Asia1980, dynia2000, marta2009, 02021968, basia1980, 07051977, 22061984, daria1986, 290980, wiki123, ola2003, marta1980, mariola, marek22, elusia1, bobinek, marta1234, Radek19, sonia1971, turek123, czubek1, 24071980

C. Poniższe hasła zostały wygenerowane przez model

Darek101, 790800, janek1977, UNALMLKK, stastazolaas, Kakan170, vizas1984, jerza19, SYOKA1234, lili1977, Alipek12, ostelus1, gabi1981, 02082001, rocka19, 51899213, Bardo1980, Qhast1234, olek123445, LOSAT19, Killo1981, franta111, jola123, 6444444, xgdvctkab, 65351322, Ciacynia, Ragon19, Tumik777, 19810172, ania2002, Tinter1234, evelalegaee, imek1963, Bajusia111, 647357710, anasser17, Gamargis, 788675605, Robandar17, Qulka19, handa100, honoser2001, nokrek1, pasiu1960, liana2007, Karbanatmf, contek1, 4428101, toria13, Zipanka1, elpas19, fimonczer1, 24091977, 7312210, zzs5715, burek109, Yatra1974, vielinek, santer19, porka2002, Wely241, dogron177, yara62, Una2008, BARAK197, Yotka1977, 9001675, soneda10, vinter19, maria1234, 94142893, vister12, Yisia12345, 51342223, ramam21, Warialao, Shomek123, ando13, Peror19, patos19, kosiek1, Olina01, kabaralaia, cantam1, 45374116, Kara123, ZAWA12344, Idakal19, 1982masio, ortala1, qwers174, TOWANA, 2307011, santer12, Tando1234, klober125, Zryra688, inta2007, 503520199, adamalaa, Xorbor2000, hontas122, Hostas1969, Marta67, 070680, tomanda111, Gaska1985, yara1985, Karasia1b1, Balina1978, 100476, deder1977, VRIIAMAN, Ulek12345, BARES1976, ulska1234, 740705, Cecanelo, Portoleaaaeea, qrka587, Qtieda1, 695661321, haluszkkek, ewa5736, 7009061, forda123, 30041958, 03061982, halek2008, Ilona1964, golqo1967, 41379570, Wertider, groberik, malara77, 10021958, Veret1234, Malos19, Ilatinek, Hanka55, unika1978, Pantakannkn, IWESA7NIIA, 43963396, Gajololnnn, Linio1974, inta1974, Kosia1957, inka1984, Cania75, 322357, 23071980, qwekw1234, Xortinska, lezio1987, fajakanegg, andon150, warka1978, reoncia, gradeasog, gred1991, Marta5757665, wieron11, Jarbolangaa, Hiasza1, Lamirustordw, Neo241209, fremina159, orka12344, Biucha12, Jagajaka, Graka1234, xawaq1, Relek1989, 021170, 2503041, lopi123444, COGONAT, 98111108, Valek1976, Mania19, ciola2001, Walna2004, ewa10011, nisia2003, hubika1, balbiszaka, urek1234, 6300000, wanda111, 09081991, Qomal1979, cieadasiek, vierek19, Pomarka, Colno100, grach123, lenio19, 82061980, donia1986, wzgbinaa, Udas1234, 220379, orta1234, qwali1973, ania2005, kasia1234, Jarkika1, ULANA1966, Anio123, zaboszek, jula1969, torian1970, 5304061, qwerna1, 1234453, sparota1, quntone1t, 6144253, Lolipasio, Amiksra, Portek1, ulek1951, 32717531, YANAUDIAAO, Urara12344, Laper1111, 00000001, Vianiomoa, 2305031, Winia1001, Zruntow1, Cujek1986, harban12, tomek1962, rawer1984, bonia1976, Tardmar, nominna115, garader, 123445666, Maryna191, 75593124, ospyczek, ramiroliaee, Varek1968, 0800081, yolka12345, Xisio7765, yekik1969, ania1978, Tara1959, SREK1995, Hania187, Jaria1985, Kencia2003, harer1980, Yola1984, TARULENAIAWAAA, Arina1983, 02krabana, hamanusikia, qwestfa, lorek1985, qwotankaaa, 6020118, 887111100, inder111, hania1981, Dionorek, qwer69, JARATANNAR, nodas81, onastartea, Unomar1s, pasia12345, atelo1956, Yiekicek, Surmor1, Qraner19, cyda1976, reginier, Romek1980, Qojka1234, Giwer1985, yulek1978, prama1981, 5270307003, Qozia1983, hanga83, alika1983, Sonia1970, quont123, Olek01,

Julia1985, Bortan12, Doras28, pioza1985, 533212598, kerans170, obik123444, ORMANE1NAKJN1, Iwolek20, Karara18, Martosn147, 150788, tomek190, IBSENA, Karandem, jatan2009, kunto19, renterinska, 03091982, ewot1980, 144444547, 51344458, Ania1977, brotek1, 67823023, jastola1, Marta1965, bobiszek1, barti1997, Lilek19, Agor1234, Qrata1978, OTZANNOLLNKNRD, imenka1, Quta123444, Risia1984, finia1979, YAJI1977, Yolek1980, leeter123, Parola1, Ulona1978, busia1234, lacek1111, USELA191, tobina12, 12345665, 24011972, 06051978, 50511551, Darina2000, ALEK12344, ILUDIA1, OLIRA19866666, 42402152, bumek19m, ramissa, jasio19, wiki1234, cuchowa, Macek620, kajalka, Kinap1983, ulga72, 58783104, Omidek1776, elasia1, 687783223, fortiler1, KARITAN1, xzj1968, Kasia1973, tanda1979, LASENA123, Bioduga, 69479301, 6791513335, ilara1980, Rowanenonmn, adagopek, 3235613, ilala1969, xsoka11, enikon1, 44825851, Kiurek1, pola1956, 050379, Daster123, Woslal1, Pelna1999, 1234305, Lamar1234, Romar84, 4043300, zema2009, Piwonia11, faral123, Odomari7ko, YARI1975, nusia54, Qoriaroeeka, OLO1973, yasia1979, caralia, Ona1981, legia1234, 5523114, 06091976, dister12, jasiek1, Yilan1202, tomasia1, xp123466, noranek, Jelikonr, Yabaja111, 42400004, Karana19, Kasia57, Norasjan, 75320009, under1973, Hidia132001, robericka, ordi1981, Jala1970, 830400ra, Samolek, elek1065, 7410021, okkiniesiae, 310681, Marta1963, BAMIS1234, 1234444, Szalo19, jalka1234, Onap12345, 02021952, Bibi1976, 76800012, xzartin1einkna, robeli11, 210894, Bioshalm, fosia836, Julia1987, Contienina, makon1980, Irubek1, 34220001, intorek1, 66618751, Parana1982, BATIKUK, 50023301, QARIA195, tomek2006, oceo1300, 070779, 31061991, Bokonka, Qziesloks, XRENKA, Regarek, Idasiora, MOGENA8N, USITA1505, Xina12344, tomarek, Camamina110, 030379, Qasiunka, forter123, czonia1, cieparcieng, faldi1987, smonki1, Kara1234, Larota1, Ziola100, torek73, 123466565, 123445736, Tandaita, 82921382, koturo1, urasz793, Xlusia201, malonlanipaea, kolek1970, marek1981, adama1234, Laczan10, Ra846129, Xasia68, ilasia1, Oletonnaae, makus200, viciulek, Olig1976, henio1971, Ewa1982, Lamla1983, 992853824, qulina1, 04041993, 8482335, donna13, qwerma1n10, mamonko, 83150963, Iwela1234, Vololy1, elota1981, yupa1984, Xlosia1, Lolka1973, Iwachtor, Cania2007, wortska, kataczek, Onika19, Iwelka1, qbina1983, 4586515, Sastik2000, tobel10, jolek1977, jola1983, 26061985, wutaga162, 30041980, tosia1980, yorand1, Taniowek, olek1977, 25051989, 86427424, Cartitas, hanga19, Sorzelea, vthwwg, Yida69, Pastusek, kamil1982, canna1989, Tomiangia, 9905521, Warta1987, werel666, agusia1, Xonta1974, Katur1972, Dalina1988, kater2003, Xula12345, redger1, Wadaplala, qesten6t1at, Inasio1, czuntan110, polek19, formoriana, 7101680, Elela12, Olek84, cezanekaea, unia519, COS2001, fitka111, Nrenonsi1r, 0000000, unia1979, Xasia1982, Kania2009, Para1971, olka1234, qwerdann, ONWA12344, 0010051, Xukasia1, Ania611, ONAN1967, kopradan, Zobiniek, Barta13, Rustat1, Disio1985, yamiki19, xika1980, rouka13, szala2003, szamower, xwipia1, Anda1978, 240679, 69764933, uricek12, 23121984, hocka1979, Doniszsko, ania1986, Ebanka1, durrialcsena, WATEK19, idek123, !nieusia1, Artinek, Xamiszkek, _anek1981, pranta1, sina84, Maratiatn1, Qafirata, Zoliteczka, Yotan1988, ania1964, 85053508, 1111111, USEK12344, CAREK1234, 40374786, ilara123, Maskalennaaiaiaaa, hera1975, Jalda1984, CARATEK, 29071974, edta300, banmar19, Piona01, 6019747, Latek1986, Eramer2004, Adata1984, robel19, 1506091, hania78, 62414032, daragoleee, 7135522, OTIKOS, marta1978, Marta31, kasia1996, emelek167, harta1980, korzi111, 51222344, ZACER1234, bartina, wara12345, 9801675, Tecia1974, 29091980, 406037153, 5167432, Geromika, Aliesioma, darasia11, Anio1972, dobim122, darek1979, yantariea, habas1234, 5047123345, 42001114, 250300, Jadrar197, Bararek, makala20003, piasse1, lenek1234, jastan123, boban1979, 310490m1, viatka169, inter1404, Kugo1976, inna2003, nonana199, horbiriae, 4168425, mikar19, olek23, alikon1110, 444445556, forwas12, kamira201, qwas12344, Bartusia1, xw1234555, Kamir56, Lena1976, olek111, Konka1234, 11051982, tomek19, Darek1979, Xzierrek, Wermara, gonia1952, 73461201, zajik1234, roman140, yarkinek, olana2003, 00550057, qwelsineekak, Drosia1, zada210, neo24marioek, Kalifa1, 47660100, Piwanek1, 91054688, Qlajka173, 19890412, 9605051005, Piowo111, pinka1973, Qhater2009, ALAN2007, Onacona, xlek32, 60313124, 18001977, Habos1985, xeszust1, laska1965, wierinaa, rubicha, Jola160, oluta1234, 9121071, Zosia1972, Zaber67, LORA2004, Caga1983, olek1300, qwel1943, leesia1, Rilaninek, Jarek1973, Minio111, yus1985, ZARASLENA, STOSANIOAKDS1, Nolo1976, flatek138, 6830001003, Naspiarekkkeak, molbast1kaale, x3rotgio, 41728100, ula2006, 69984989, 6011003, tamana1971, Santono, Emironto, dararka, 19811105s, Karanomitueaaa, Jalimobaa, syndienka, 43589707, HANOSNA, 200361, Atusia1, 42336444, ewa9123, CUZA1234, Elatola, darek1994, ENA1964, Morto19, 8507285, neo46664, lolka1979, wines1983, dosdarek, ciosza1, Uricha1, Qrasta18, Ctelek111, Yukos155, 68293863, Jalila12, ikiter19, lenia1234, Janka1234, czeda1985, 69041000, Hawastattkgn, Hania1965, czaleta, Zafina1, ulia1972, Danka1970, synek1977, Yiper1974, Tuga1971, jatero05, BACIA197, Lindas1, monio1982, ZAN1111, ZEMADAL, Xoxart1le, qlana102, 4105101, yomek1988, qocho1985, yantaleao, Marpamiki, DARURAK16, forwal1, harwiana, asia167, UTAKA1234, 27071970, zola12344, Iviola1, Halek1982, ordisina, 300000101, 400134203, gresza11k1, Vojcio11, ewera1985, xricha1, golika1n1211, Renbakea1, bolo12345, Jargant1, 7630001, adik1951, Pasiek170, leti1234, Wister19, 04102009, Rudusia1, Gorka1234, qwodnied, Xzurek11, Zadstard, Daras2002, Barka67, aniusia, Dorek2007, Kamisia211, 7608010, Tigrymia, Vuatimas, tomek1984, 21021976, Titka2005, romek1979