Metody Redukcji Wymiarów Dokumentów Tekstowych Przy Użyciu Probabilistycznych Modeli Graficznych

mgr Maciej Jankowski

Promotor: prof. dr hab. inż. Marian Chudy

Wydział Cybernetyki
Wojskowa Akademia Techniczna w Warszawie
maciej.jankowski@wat.edu.pl

1 Przedmiot i zakres rozprawy

Jednym z poważnych problemów współczesnej analizy danych jest to, że dane charakteryzują się dużą liczbą wymiarów. Problem ten jest szczególnie poważny podczas analizy danych tekstowych. Popularne reprezentacje danych tekstowych, macierze TF oraz TF-IDF, mają wymiar równy liczbie słów w słowniku. W praktycznych zastosowaniach, liczba ta często przekracza kilkaset tysięcy.

Tak duża liczba wymiarów powoduje problemy podczas estymacji parametrów modeli uczenia maszynowego. Do najczęstszych problemów należą: akumulacja szumu, korelacja pomiędzy wymiarami oraz wysoka złożoność obliczeniowa, zarówno czasowa jak i pamięciowa. Z tego powodu, przed przystąpieniem do analizy danych np. regresji czy klasyfikacji, dane często przechodzą proces redukcji wymiarowości. W przypadku dokumentów tekstowych, jednym ze sposobów zmniejszenia wymiarowości jest znalezienie ukrytej reprezentacji danych zwanej tematami. Najpopularniejszy obecnie algorytm, który pozwala znaleźć te tematy to Latent Dirichlet Allocation (LDA). Model LDA, jest jednym z przykładów szerszej grupy modeli zwanych Probabilistycznymi Modelami Graficznymi. W tego typu modelach, mamy z reguły trzy rodzaje elementów: zmienne które obserwujemy (np. dokumenty), zmienne których nie możemy bezpośrednio zaobserwować (np. tematy), oraz parametry. Zmienne, których bezpośrednio nie obserwujemy, będziemy nazywać zmiennymi ukrytymi. Mimo iż zmienne te nie mogą być bezpośrednio zaobserwowane, staramy się opisać je w sposób statystyczny, estymując parametry ich rozkładów.

Aby móc skorzystać z algorytmu LDA lub Multi-class sLDA (nadzorowana wersja algorytmu LDA), musimy wybrać liczbę tematów K jeszcze przed rozpoczęciem trenowania modelu. Metody proponowane w literaturze [6], [4], [2] oraz [5], często działają tylko dla pewnych zbiorów danych i nie sprawdzają się dla innych. Dodatkowo, w przypadku klasyfikacji na danych wygenerowanych przez model LDA, wybór najlepszej liczby tematów, może nie być możliwy przy użyciu tylko nienadzorowanych metod. W pracy zaproponowałem kilka metod na

oszacowanie tej liczby, oraz na uniknięcie wyboru pojedynczej liczby K, poprzez zastosowanie ensemble modeli.

Algorytmy oparte o LDA, wymagają sporo obliczeń, dlatego ich zastosowanie do dużych zbiorów danych jest ograniczone. W rozprawie, zostały przeanalizowane metody oparte o najnowsze osiągnięcia z dziedziny Bajesowskich Sieci Neuronowych, które pozwalają zredukować wymiar danych w dużo krótszym czasie niż algorytmy oparte o LDA. Zostały zaproponowane rozszerzenia bardzo popularnego algorytmu Variational Autoencoder (VAE). Implementacje tych metod, korzystają z popularnego narzędzia do budowy sieci neuronowych Tensorflow [1]. To narzędzie, może być uruchamiane zarówno na procesorach CPU jak i GPU, bez żadnej zmiany kodu źródłowego. Wykorzystanie infrastruktury GPU, może jeszcze bardziej przyczynić się do zmniejszenia czasu potrzebnego na wykonanie obliczeń.

2 Cel rozprawy

Główne cele rozprawy to:

1. Przebadanie obecnych, nienadzorowanych metod dotyczących redukcji wymiarów w analizie dokumentów tekstowych, w szczególności: Latent Semantic Analysis (LSA) [11], Probabilistic Latent Semantic Analysis (PLSA) [7], Latent Dirichlet Allocation (LDA) [3] oraz Variational Autoencoder (VAE) [10]
2. Zaproponowanie metody radzenia sobie z wyborem parametru K, oznaczającego wymiar przestrzeni zmiennych ukrytych w algorytmie LDA
3. Zaproponowanie metody radzenia sobie z wyborem parametru K dla modelu Multi-class sLDA, oraz poprawienie jakości tego klasyfikatora (zmniejszenie liczby błędnie zaklasyfikowanych obserwacji)
4. Przebadanie nadzorowanej wersji algorytmu VAE. Rozszerzenie istniejących koncepcji na klasyfikację przy użyciu Supervised Variational Autoencoder (SVAE) w taki sposób, aby przestrzeń ukryta lepiej odzwierciedlała problem klasyfikacji.

3 Struktura rozprawy

W rozdziale pierwszym zawarty został przegląd podstawowych zagadnień dotyczących redukcji wymiarów i klasyfikacji.

W rozdziale drugim, zostały przeanalizowane obecnie istniejące metody znalezienia parametru K (wymiar przestrzeni ukrytej, liczba tematów). Następnie, wykonane zostały eksperymenty z dwiema nowymi metodami wyboru tego parametru. Pierwsza metoda, polega na obliczeniu średniej entropii tematów i wyborze modelu z najniższą entropią. Jeżeli tematy danego dokumentu mają niską entropię, to możemy się spodziewać, że dokument jest charakteryzowany przez klika tematów o wysokim prawdopodobieństwie. Jeżeli rozkład tematów w dokumencie ma wysoką entropię, to dokument zawiera wiele tematów w mniej więcej

podobnych proporcjach. Na przykładzie zbioru danych Film Reviews, można zauważyć, że entropia jest skorelowana z poziomem błędu. Kolejny eksperyment na przykładzie tego samego zbioru danych pokazał, że najlepsza liczba tematów K dla modelu LSA w kontekście klasyfikacji, może być dobrym wyborem parametru K dla LDA. W drugiej części pracy, zaproponowałem trzy metody oparte o ensemble modeli, które pozwalają uniknąć wyboru pojedynczej wartości dla K. Zamiast tego, budujemy klasyfikator złożony z modeli o różnej liczbie tematów. W ostatniej sekcji tego rozdziału, przedstawione zostały wyniki eksperymentów. Wyniki zaprezentowane w rozdziale drugim, pierwszy raz zostały przebadane i opublikowane w pracy [9].

W rozdziale trzecim, została zaprezentowana koncepcja rozszerzenia algorytmu Multi-class sLDA poprzez zbudowanie ensemble tych modeli. Taki model, nie wymaga wyboru pojedynczego parametru określającego liczba tematów. Dodatkowo, ponieważ model ten zawiera wiele modeli Multi-class sLDA, będzie on w stanie lepiej dopasować się do dokumentów zawierających różną liczbę tematów. Wykazałem, że dla zbiorów danych SmsSpam oraz Poliblog, zastosowanie ensemble do Multi-class sLDA, daje lepsze wyniki, niż ręczny wybór parametru określającego liczbę tematów. Na pierwszym zbiorze danych (SmsSpam), udało nam się zredukować poziom błędu z 18% do 12%. Na drugim zbiorze danych, nowa metoda obniżyła błąd z 29% do 22%. Wyniki zaprezentowane w rozdziale trzecim, pierwszy raz zostały przebadane i opublikowane w pracy [8].

Powyższe metody, wymagają wielokrotnego trenowania modeli LDA. Jest to procedura czasochłonna. Dlatego w kolejnym rozdziale, przebadane zostały metody, pozwalające w szybszy sposób dokonać redukcji wymiarów.

W czwartym rozdziale, główny nacisk został położony na przebadanie i zaproponowanie rozszerzeń do metod opartych o najnowsze osiągniecia w obszarze Bajesowskich Sieci Nauronowych. W pierwszej części, wykonałem testy redukcji wymiarów przy użyciu algorytmów Variational Autoencoder (VAE) oraz Supervised Variational Autoencoder (SVAE), na pięciu zbiorach danych. Następnie, zostały opracowane i przebadane dwie metody będące rozszerzeniem SVAE. Metody te zostały nazwane DLGMM oraz DSLGMM. Zaproponowałem modyfikacje funkcji celu, które traktują zmienne ukryte, jak gdyby pochodziły z mieszanki rozkładów Gaussowskich. W pierwszym przypadku, zmienne ukryte są znajdowane w sposób nienadzorowany, tzn. nie dostarczamy informacji o przypisaniu wartości zmiennej ukrytej do konkretnego modelu Gaussowskiego. W drugim przypadku, nadzorujemy proces znajdowania tych zmiennych, poprzez jawne przypisanie każdej obserwacji do jednego rozkładu. To przypisanie jest z reguły pewną permutacją etykiet obserwacji. Zaprezentowane zostały również dwa nowe zastosowania tych modeli. Pierwsze zastosowanie, to score klasyfikatora oparty o obliczenie normy wartości ukrytej zmiennej losowej $||z||$. Porównałem wyniki z naturalnym scorem $p(c|w)$ i wykazałem, że nasza metoda daje lepsze rezultaty dla pięciu zbiorów danych. Drugie zastosowanie polega na możliwości losowania danych z jednej, z góry ustalonej, klasy.

4 Ensemble Modeli LDA w celu uniknięcia wyboru pojedynczej liczby tematów

Aby uniknąć wyboru pojedynczej liczby tematów, przebadaliśmy możliwości zastosowania trzech metod połączenia modeli w ensemble.

Pierwszą z zastosowanych metod, jest głosowanie większościowe. Na etapie predykcji, każdy model zwraca najbardziej prawdopodobną klasę. Ta klasa, która otrzymała największą liczbę głosów wygrywa. Niech $S_l^{MV} : W^+ \rightarrow \{0,1\}^C$, będzie funkcją pojedynczego klasyfikatora, zwracającą wektor *one-hot*. Tylko jeden element tego wektora ma wartość 1, a wszystkie pozostałe elementy mają wartość 0. Niech $F_d^{MV} \in \{0,1\}^{L\times C}$, będzie macierzą zawierającą te wektory. Macierz jest zdefiniowana dla jednego dokumentu, L klasyfikatorów oraz C klas. Pojedynczy element $(F_d^{MV})_{lc}$, jest równy 1, jeśli model l wybrał klasę c. W przeciwnym przypadku jest równy 0. Klasyfikator oparty o głosowanie większościowe jest zdefiniowany jako

$$E^{MV}(w^{(d)}) = \underset{c\in\{1,\ldots,C\}]}{\operatorname{argmax}} \sum_{l=1}^{L} (F_d^{MV})_{lc}.$$

Druga metoda, głosowanie większościowe z wagami, jest bardzo podobna do pierwszej metody, ale zamiast liczyć, ile głosów zostało oddanych na każdą z klas, liczona jest średnia ważona tych głosów. Musimy tutaj założyć, że każdy klasyfikator zwraca funkcję *score*, która jest tym wyższa, im klasyfikator jest bardziej pewny swojej predykcji. Niech S_l^{MVW} będzie wektorem liczb z przedziału $(0,1)$, oraz $F_d^{WMV} \in (0,1)^{L\times C}$ będzie macierzą predykcji, zawierającą L wierszy i C kolumn. Wówczas, pojedyncza wartość $(F_d^{WMV})_{lc}$, opisuje jak bardzo model l jest pewny, że poprawna klasa to c . Klasyfikator oparty o głosowanie większościowe z wagami definiujemy jako

$$E^{WMV}(w^{(d)}) = \underset{c\in\{1,\ldots,C\}]}{\operatorname{argmax}} \sum_{l=1}^{L} (F_d^{WMV})_{lc}.$$

Ostatnia z metod, wykorzystuje funkcję perplexity [3], która jest bardzo popularną funkcją służącą do oszacowania modelu języka. Niech $p_l(w^{(d)})$, będzie prawdopodobieństwem dokumentu $w^{(d)}$ w modelu $\mathcal{M}_l$. Perplexity definiujemy jako

$$P_{\mathcal{M}_l}(w^{(d)}) = \exp\left\{ -\frac{\log p_l(w^{(d)})}{N_d} \right\}.$$

Funkcja ta została użyta w następujący sposób, dla każdego dokumentu $w^{(d)}$, obliczone zostało perplexity w każdym z modeli. Model, dla którego perplexity było najniższe, został użyty do predykcji klasy dokumentu $w^{(d)}$

$$E^{PERP}(w^{(d)}) = \mathcal{M}_{g(w^{(d)})}(w^{(d)}),$$

gdzie

$$g(w^{(d)}) = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmin}} P_{\mathcal{M}_l}(w^{(d)}).$$

W ten sposób, używamy tylko jednego modelu dla każdego dokumentu, ale każdy dokument może używać jednego z wielu modeli. Wyniki zostały zaprezentowane w tabelach 1 (Mobile Apps) oraz 2 (Poliblog).

Tablica 1: Porównanie błędów klasyfikacji pojedynczego modelu oraz ensemble modeli dla zbioru Mobile Apps

Metoda	Poziom błędu
Najlepszy pojedynczy model (50 topics)	0.3368421
Głosowanie większościowe	0.3368421
Głosowanie większościowe z wagami	0.3236842
Perplexity driven ensemble	0.2971429

Tablica 2: Porównanie błędów klasyfikacji pojedynczego modelu oraz ensemble modeli dla zbioru Poliblog

Metoda	Poziom błędu
Best single model (50 topics)	0.285
Majority voting	0.270
Weighted majority voting	0.265
Perplexity driven ensemble	0.275

5 Nowy algorytm oparty o ensemble wielu modeli Multi-class sLDA

Aby zbudować ensemble wielu modeli Multi-class sLDA, wykorzystany został algorytm AdaBoost. Jest to algorytm iteracyjny, który w każdej nowej iteracji modyfikuje wagi przykładów uczących w taki sposób, aby przykłady błędnie zaklasyfikowane w poprzedniej iteracji, miały w nowej iteracji większą wagę.

Załóżmy, że zmienna losowa y przyjmuje wartości w zbiorze $\{0, 1\}$. Każdy klasyfikator Multi-class sLDA $\mathcal{M}_l : \{0,1\}^{N_d \times V} \to \{-1, 1\}$, definiujemy jako

$$\mathcal{M}_l(w^{(d)}) = \underset{c \in \{-1,1\}}{\operatorname{argmax}}\, p(Y = c | w^{(d)}).$$

Boost Multi-class sLDA, $\mathcal{M} : \{0,1\}^{N_d \times V} \to \{-1,1\}$, to ensemble L modeli Multi-class sLDA, który definiujemy jako

$$\mathcal{M}(w^{(d)}) = \text{sign}\left(\sum_{l=1}^{L} \alpha_l \mathcal{M}_l(w^{(d)})\right),$$

gdzie wagi $\alpha_1, \ldots, \alpha_L$ są wyznaczone przy użyciu algorytmu 1. Na rysunku 1,

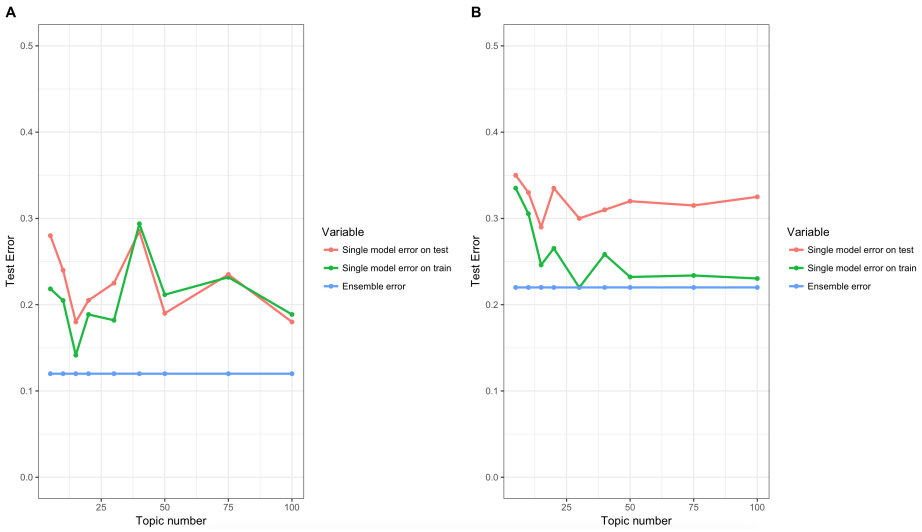
Algorithm 1 Boost Multi-class sLDA

1. Zainicjuj wagi pierwszego modelu $r^{(1)} \in \mathbb{R}^M$, $r_d^{(1)} = 1/M$, $d = 1, 2, \ldots, M$
2. For $l = 1$ to L:
   (a) Wytrenuj Multi-class sLDA model $\mathcal{M}_l$, w oparciu o dane treningowe z wagami $r^{(l)}$, używając algorytmu Variational Expectation Maximization (VEM). Szczegóły dotyczące obliczania parametrów w kroku E oraz M są podane w pracy.
   (b) Oblicz
   $$\text{err}_l = \frac{\sum_{d=1}^{M} r_d^{(l)} I[c_d \neq \mathcal{M}_l(w_d)]}{\sum_{d=1}^{M} r_d}.$$
   (c) Oblicz $\alpha_l = \log((1 - \text{err}_l)/\text{err}_l)$.
   (d) Uaktualnij wagi obserwacji $r_d^{(l+1)} \leftarrow r_d^{(l)} \cdot \exp[\alpha_l \cdot I[c_d \neq \mathcal{M}_l(w_d)])]$, $d = 1, 2, \ldots, M$.
3. Wynikiem algorytmu jest model
$$\mathcal{M}(w_d) = \text{sign}\left(\sum_{l=1}^{L} \alpha_l \mathcal{M}_l(w_d)\right),$$

przedstawiamy porównanie algorytmu Boost Multi-class sLDA z algorytmem Multi-class sLDA dla różnych wartości parametru K.

6 Algorytm nadzorowany oparty o Wariacyjny autoenkoder

Wariacyjny autoenkoder, jest graficznym modelem probabilistycznym z ukrytymi zmiennymi, którego parametry są uczone przy wykorzystaniu sieci neuronowej. Jest to obecnie bardzo atrakcyjny algorytm, ponieważ jego implementacja, nie wymaga wykonania skomplikowanych obliczeń rachunkowych. Dodatkowo, parametry wariacyjne, są dobierane łącznie dla wszystkich dokumentów z korpusu, a nie jak w przypadku modelu LDA, oddzielnie dla każdego dokumentu. W oryginalnej formie, wariacyjny autoenkoder jest metodą nienadzorowaną. Możemy użyć tego modelu, do znalezienia niskowymiarowej reprezentacji naszych danych. Następnie dokonać klasyfikacji na przekształconych danych. Ponieważ



Rysunek 1: Porównanie poziomu błędu modelu Multi-class sLDA dla różnej liczby tematów oraz Boost Multi-class sLDA. Po lewej stronie zbiór SmsSpam (A), po prawej stronie zbiór Poliblog (B)

model ten jest nienadzorowany, możemy napotkać na podobne problemy jak w przypadku modelu LDA. Dlatego, w pracy doktorskiej, zostały zaprezentowane koncepcje algorytmów nadzorowanych, które pozwalają uzyskać niskowymiarową reprezentację danych w taki sposób, aby maksymalizowała ona jakość klasyfikacji. W pracy zostały zaproponowane modele DLGMM oraz DSLGMM.

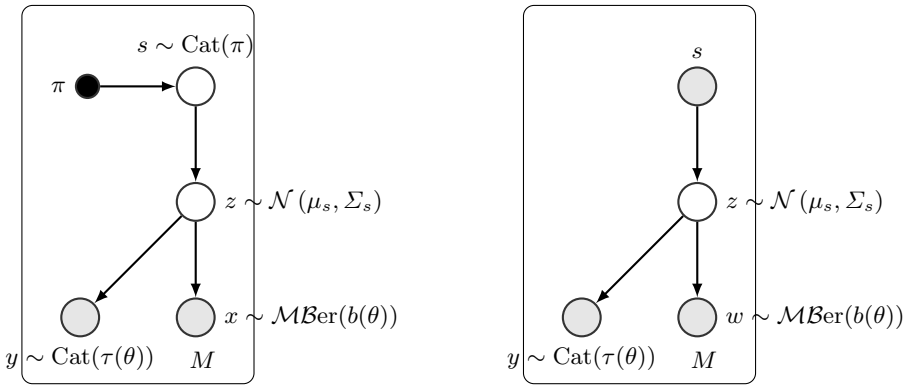
Niech C będzie liczbą klas, P, R oraz J będą odpowiednio wymiarami dokumentów, klas oraz zmiennych ukrytych. Niech M będzie liczbą obserwacji (dokumentów). Zmienne losowe $w \in \{0,1\}^P$, $y \in \{0,1\}^R$ and $z \in R^J$ oznaczają dokument, klasę oraz ciągłą zmienną ukrytą. Niech $p(w,z)$ będzie modelem parametrycznym rozkładu łącznego zmiennych w oraz z. Proces generujący dla pierwszego z modeli DLGMM jest następujący

1. Wylosuj $s^{(d)} \sim \text{Cat}(\pi)$
2. Wylosuj $z^{(d)} \sim \mathcal{N}(\mu_{s^{(d)}}, \Sigma_{s^{(d)}})$
3. Wylosuj $w^{(d)}|z^{(d)}, \theta \sim \mathcal{MB}\text{er}(b(\theta))$,
4. Wylosuj $y^{(d)}|z^{(d)}, \theta \sim \text{Cat}(\tau(\theta))$,

Proces generujący dla drugiego z modeli to

1. Wylosuj $z^{(d)}|s^{(d)} \sim \mathcal{N}(\mu_{s^{(d)}}, \Sigma_{s^{(d)}})$
2. Wylosuj $w^{(d)}|z^{(d)}, \theta \sim \mathcal{MB}\text{er}(b(\theta))$,
3. Wylosuj $y^{(d)}|z^{(d)}, \theta \sim \text{Cat}(\tau(\theta))$,

Rysunek 2 przedstawia graficzne reprezentacje zaproponowanych modeli. Na rysunku 3, zostały przedstawione wyniki klasyfikacji na dwuwymiarowych danych



Rysunek 2: Graficzna prezentacja procesów generujących modeli DLGMM oraz DSLGMM

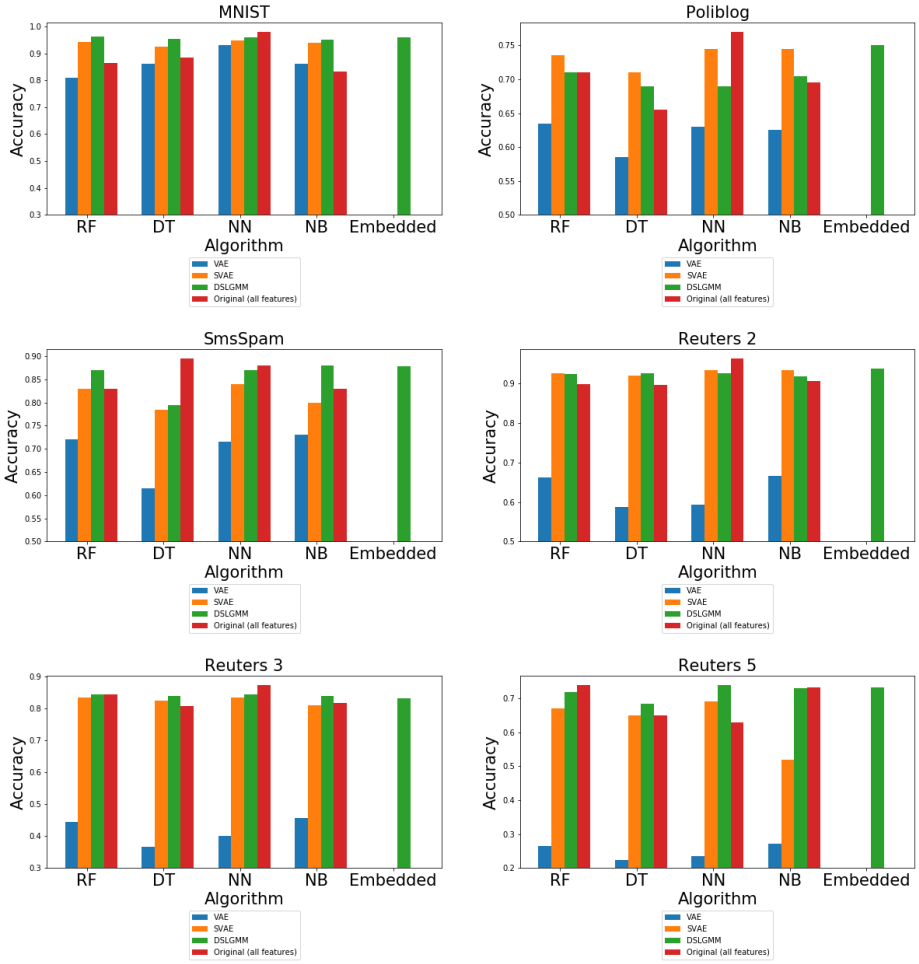
.

wygenerowanych ze zmiennych ukrytych z modeli VAE, SVAE, DSLGMM oraz na oryginalnych danych. Na osi pionowej, znajduje się dokładność klasyfikatora (liczba poprawnie zaklasyfikowanych przykładów podzielona przez liczbę wszystkich przykładów) Jak łatwo możemy zaobserwować, wyniki na danych wygenerowanych przez model DSLGMM, są znacznie lepsze niż na tych wygenerowanych przez VAE.

Głównym problemem modeli opartych o LDA oraz Multi-class sLDA jest czas potrzebny na obliczenia. W rozdziale czwartym zostało wykazane, że model DSLGMM działa znacznie szybciej niż Boost Multi-class sLDA. Wszystkie eksperymenty zostały przeprowadzone na laptopie MacBook Pro z procesorem 2,2 GHz Intel Core i7 oraz pamięcią operacyjną 16 GB 1600 MHz DDR3.

Dla zbioru danych SmsSpam, pojedyncza iteracja modelu Multi-class sLDA zajmuje ok. 11 sekund. Zgodnie z tym co zostało zaprezentowane w rozdziale trzecim, 35 iteracji algorytmu Boost Multi-class sLDA pozwala uzyskać błąd na poziomie 12%. Oznacza to, że model potrzebuje ok. 6 minut i 40 sekund na obliczenia. Dla porównania, trenowanie algorytmem DSLGMM, zajmuje 5 sekund i osiąga podobny poziom błędu (zobacz tabele 3 oraz 4).

Różnice te są jeszcze bardziej widoczne dla zbioru danych Poliblog. Algorytm Boost Multi-class sLDA potrzebuje ok. 25 iteracji, aby osiągnąć poziom błędu równy 25%. To wymaga 2 godzin i 18 minut obliczeń. Dla porównania, algorytm DSLGMM potrzebuje tylko 5 sekund na wytrenowanie modelu do poziomu błędu 25%. Jednakże, przewagą algorytmu Boost Multi-class sLDA jest to, że kolejne iteracje jeszcze bardziej obniżają poziom błędu, ostatecznie osiągając błąd 22% po 63 iteracjach. To jednak, zajmuje 5 godzin i 48 minut (zobacz tabele 3 oraz 4).



Rysunek 3: Porównanie jakości klasyfikacji modeli VAE, SVAE and DSLGMM.

Tablica 3: Porównanie czasów trenowania dla zbiorów SmsSpam i Poliblog

Dataset	Boost sLDA (jedna iteracja)	Boost sLDA	DSLGMM (5 epochs)
SmsSpam	11 sec.	6 min (35 iter)	5 sec.
Poliblog	5.4 min.	5.8 hours (63 iter) 2 hours 18 min. (25 iters)	5 sec.

Tablica 4: Porównanie poziomu błędu dla zbiorów SmsSpam i Poliblog

Dataset	Boost sLDA	DSLGMM (5 epochs)
SmsSpam	0.12 (35 iter)	0.12
Poliblog	0.22 (63 iter) 0.25 (25 iter)	0.25

7 Rezultaty uzyskane w rozprawie

Główną myślą rozprawy było zbadanie metod redukcji wymiarów dokumentów tekstowych w kontekście klasyfikacji. W trakcie analizy istniejących metod, zauważyłem szereg problemów i możliwości usprawnień, które stały się inspiracją do zbudowania nowych metod. W rozprawie, zostały zaproponowane następujące nowe techniki badawcze oraz algorytmy:

- W rozdziale drugim, zostały zaproponowane dwie metody na oszacowanie liczby tematów w algorytmie LDA. W tym samym rozdziale, zostały też przebadane trzy metody łączenia algorytmów, które pozwalają uniknąć problemu wyboru pojedynczej liczby określającej liczbę tematów w modelu LDA
- W rozdziale trzecim, została zaprezentowana metoda łączenia algorytmów Multi-class sLDA w taki sposób, aby nie trzeba było wybierać jednej liczby dla parametru K. Zostało również wykazane, że metoda ta, osiąga znacznie niższy błąd klasyfikacji, niż pojedynczy model Multi-class sLDA.
- W rozdziale czwartym, zaproponowałem rozszerzenia algorytmu VAE w taki sposób, aby uczenie odbywało się w sposób nadzorowany, oraz żeby przestrzeń ukryta lepiej odzwierciedlała różnice i podobieństwa pomiędzy klasami. To pozwala uzyskać bardzo dobre wyniki klasyfikacji. Dodatkowe zalety naszego modelu, to możliwość generowania przykładów dla jednej, z góry wybranej, klasy. Został również zaproponowany lepszy score dla modelu DSLGMM niż naturalny score $p(y|w)$.

Przeprowadzone w ramach niniejszej rozprawy doktorskiej badania potwierdziły, że możemy z powodzeniem dokonywać redukcji wymiarów w dokumentach tekstowych przy użyciu probabilistycznych modeli graficznych. Wykazane zostało również, że jakość istniejących modeli może zostać znacznie usprawniona poprzez zastosowanie nowatorskich metod opisanych w pracy.

Literatura

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015. Software available from tensorflow.org.
2. Rajkumar Arun, Venkatasubramaniyan Suresh, CE Veni Madhavan, and MN Narasimha Murthy. On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 391–402. Springer, 2010.
3. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

4. Juan Cao, Tian Xia, Jintao Li, Yongdong Zhang, and Sheng Tang. A density-based method for adaptive lda model selection. *Neurocomputing*, 72(7-9):1775–1781, 2009.
5. Romain Deveaud, Eric SanJuan, and Patrice Bellot. Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique*, 17(1):61–84, 2014.
6. Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.
7. Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.
8. Maciej Jankowski. *Boost Multi-class sLDA Model for Text Classification.*, page 633–644. Springer International Publishing AG, 2018.
9. Maciej Jankowski. Ensemble methods for improving classification of data produced by latent dirichlet allocation. *Computer Science and Mathematical Modelling*, (8):17–28, 2018.
10. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
11. Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.