DriftLens: A Concept Drift Detection Tool

Demonstration Paper

DriftLens: A Concept Drift Detection Tool

Salvatore Greco
Politecnico di Torino
Turin, Italy
salvatore_greco@polito.it

Bartolomeo Vacchetti
Politecnico di Torino
Turin, Italy
bartolomeo.vacchetti@polito.it

Daniele Apiletti
Politecnico di Torino
Turin, Italy
daniele.apiletti@polito.it

Tania Cerquitelli
Politecnico di Torino
Turin, Italy
tania.cerquitelli@polito.it

ABSTRACT

Concept drift refers to changes in data distribution over time that can lead to performance degradation of deep learning systems. Production models need to be continuously monitored for drift. Detecting concept drift poses significant challenges for deep classifiers working with unstructured data, especially when the true labels for new samples are not available and the data has high dimensionality. In such scenarios, drift detection must be approached using unsupervised methods.

This paper presents the demo of a tool that uses an effective unsupervised drift detection technique for deep classifiers on unstructured data, namely DriftLens. The tool enables users to i) experiment with different controlled drift patterns on multiple preloaded text and image classifiers and ii) detect possible drifts on new models and data streams. The recorded demo of the tool, available at https://youtu.be/1R2igFhMD8U, shows how end users can interact with DriftLens and use it to continuously monitor models for concept and data drift.

1 INTRODUCTION

Concept drift refers to unpredictable changes in the underlying distribution of data streams over time [15] caused by the dynamic nature of real-world systems. When a model processes a data stream that is subject to concept drift, performance degradation can occur. Therefore, it is crucial to continuously monitor models in production systems to recognize when this phenomenon occurs and take action to restore the model's original performance.

Detecting concept drift presents significant challenges in classification problems when the true labels for newly processed samples are not available in the data stream, and therefore, only the predicted values are available. In such scenarios, performance-based drift detectors [1], which directly measure the performance degradation of the model, i.e., the difference between the true labels and the model predicted values, are not applicable due to the lack of the true labels. Instead, unsupervised concept drift detectors must be applied [10, 13].

Most unsupervised drift detection methods focus on examining the new data distributions or distances to identify possible drifts [2, 3, 6, 7, 16, 22]. However, most of these techniques are inaccurate or too complex for real-time drift detection [23]. These limitations are even more pronounced with classifiers working on unstructured data due to the high input dimensionality and lack of a fixed data structure, such as columns.

DriftLens [11] is an effective unsupervised drift detection technique suitable for such scenarios. It is based on the distribution distances of embedding representations generated by deep learning models on unstructured data. The distribution distances are calculated between fixed-size windows of the new data stream and a historical dataset called baseline. If the distribution distances exceed certain thresholds, drift is predicted. DriftLens also performs a characterization of the drift by identifying the labels most affected by the drift. Due to its low computational complexity, it allows real-time drift monitoring. The main features of the DriftLens methodology are summarized in §2.

This paper presents a demo of a tool based on the DriftLens methodology (§3) that enables users to i) experiment with concept drift detection on different text and image classifiers and controlled drift patterns on a set of pre-uploaded use cases and ii) detect possible drift in new models and datastream.

The code implementation is freely accessible[1] to facilitate the use of the tool. A recorded demo showing how users can interact with the tool is also available at https://youtu.be/1R2igFhMD8U.

2 DRIFTLENS METHODOLOGY

DriftLens [11] is an *unsupervised* drift detection technique based on distribution distances within the embedding representations generated by deep learning models when working with unstructured data. It computes the distances between the baseline distribution, i.e., the distribution of the embedding representation extracted from a historical dataset, and the embedding distribution of new samples divided into fixed-size windows, as summarized in Figure 1. Note that this technique is *unsupervised* and does not require the true labels in the new data stream. Due to its low complexity, it can perform drift detection in real-time. The methodology includes an *offline* and an *online* phase. However, the data modeling and distribution distances are calculated similarly in both phases.

Data modeling. A given batch of data (i.e., the entire baseline or a new window) is modeled by estimating the multivariate normal distribution of the embedding vectors. Specifically, the distribution is represented by the embeddings' mean vector and covariance matrix.

Distribution distance. The Frechét Inception Distance (FID) score is used to compute the distance between two multivariate normal distributions [9]. It is a real $[0, \infty]$ range value. The higher the score, the greater the distance and the more likely the drift.

[1] GitHub repository: https://github.com/grecosalvatore/DriftLensDemo

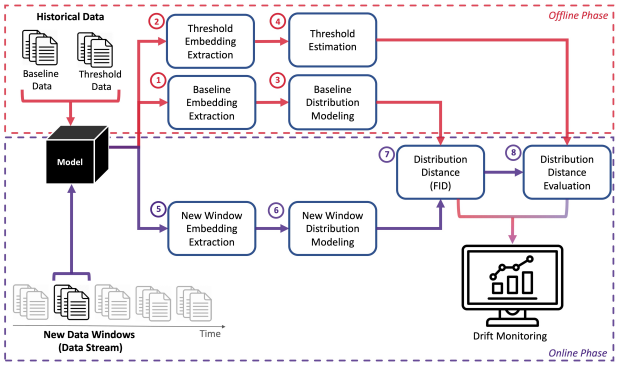

Figure 1: DRIFTLENS **architecture.**

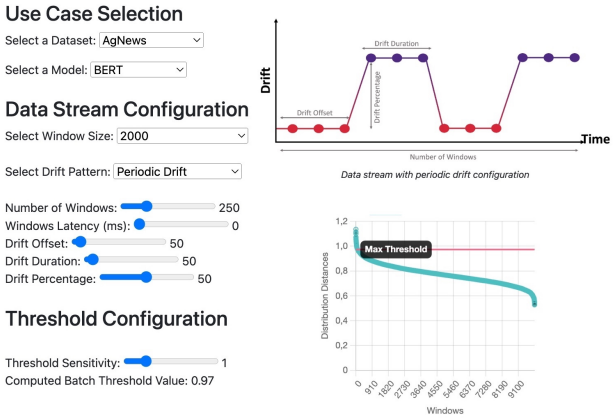

Figure 2: Controlled drift experiment configuration.

2.1 Offline phase

In the *offline* phase, DRIFTLENS estimates the distributions of a historical dataset, called baseline, which represents the concepts that the model has learned during training. It then estimates thresholds that discriminate between normal and abnormal (i.e., potential drift) distribution distances. For example, the baseline and threshold datasets can be the training and test set, respectively, or a set of data from the new data stream where it is known that drift is not present. In our experiments, we used the training and test sets as baseline and threshold datasets. From these datasets, the embedding representations and the predicted labels are extracted using the deep learning model (Figure 1-① and ②).

Baseline distribution modeling. The baseline dataset is used to model what we call the *per-batch* and *per-label* multivariate normal distributions (Figure 1-③). The *per-batch* models the entire vector distributions independently of the class label. The mean and covariance are computed on the entire set of embeddings. Instead, for the *per-label*, n normal distributions are estimated, where n is the number of labels the model was trained on. Each one is modeled by grouping the embeddings by predicted labels and computing the mean and covariance of each label separately.

Threshold estimation. The threshold dataset is used to estimate the distance thresholds (Figure 1-④). Thresholds are used in the *online* phase to discriminate between normal and abnormal (i.e., potential drift) distribution distances. To compute the threshold, a large number of windows are sampled from the threshold dataset. The window size is the same as in the *online* phase. The distances of the *per-batch* and the *per-label* distribution are then calculated between the baseline and each window, and sorted in descending order. The maximum value in this list of distances represents the maximum distance of a set of samples considered without drift. The idea behind this is that if a new value for the distribution distance of a window exceeds the threshold value, this represents a possible drift. For this reason, DRIFTLENS calculates an overall *per-sbatch* threshold, and a *per-label* threshold for each class.

2.2 Online phase

In the *online* phase, DRIFTLENS analyzes the new data stream in fixed-size windows. For each window, the process is similar to the *offline* phase. Firstly, the embedding vectors and the predicted labels are produced by the model (Figure 1-⑤). Secondly, the *per-batch* and *per-label* distributions are modeled by computing the mean and covariance of the embedding for the entire window and the samples predicted for each label separately (Figure ⑥). Then, the *per-batch* and *per-label* distribution distances between the embeddings of the current window and the baseline are computed (Figure 1-⑦). Finally, if the distribution distances exceed the threshold values, drift is predicted for the current window (Figure 1-⑧).

The process is repeated for each window. A drift monitor plots the distribution distances *per-batch* and *per-labels* separately over time. When drift is predicted (i.e., the distribution distances exceed the threshold values), it is indicated in the plot (see §3.2).

Table 1: Pre-uploaded use cases for controlled experiments.

Dataset	Use Case	Models	F1	Description
Ag News (Text)	1.1 1.2 1.3	BERT DistillBERT RoBERTa	0.98 0.97 0.98	**Task**: Topic Classification. **Training Labels**: *World, Business, Sport* **Drift**: *Science/Tech*
20 News (Text)	2.1 2.2 2.3	BERT DistillBERT RoBERTa	0.88 0.87 0.88	**Task**: Topic Classification. **Training Labels**: *Technology, Sale-Ads, Politics, Religion, Science* **Drift**: *Recreation*
Intel (Image)	3.1 3.2	VGG16 ViT	0.89 0.90	**Task**: Image Classification **Training Labels**: *Forest, Glacier, Mountain, Building, Street* **Drift**: *Sea*
STL-10 (Image)	4.1 4.2	VGG16 ViT	0.82 0.96	**Task**: Image Classification **Training Labels**: *Airplane, Bird, Car, Cat, Deer, Dog, Horse, Monkey, Ship* **Drift**: *Truck*

3 DRIFTLENS TOOL

The tool is a web application implemented in Flask [12] based on the DRIFTLENS methodology. The home page describes what users can do and provides a link to detailed documentation of the concept drift problem and the DRIFTLENS methodology. Two pages are available to launch the experiments. The first page enables users to select a previously uploaded use case and generate a data stream containing a controlled generated drift (§3.1). The second page enables users to perform drift detection on their own dataset and model (§3.3). In both scenarios, a drift monitor indicates whether a drift is detected (§3.2).

3.1 Run controlled drift experiments

The first page enables users to select a pre-uploaded use case to configure a controlled drift experiment, to which DriftLens will be applied (Figure 2).

Pre-uploaded use cases. Table 1 summarizes all the available use cases. Currently, the tool provides four pre-uploaded datasets: Ag News [24] and 20 Newsgroups in NLP, Intel Image [17] and STL-10 [4] in computer vision. For the NLP datasets, the BERT [5] RoBERTa [14], and DistilBERT [18] models were trained. Instead, for the compute vision datasets, the VGG16 [19] and Vision Transformer (ViT) [8] ones. As a result, the user can experiment with 10 possible use cases. We designed the following settings to simulate drift. We used a subset of the dataset to train and test the models. A part of the dataset is kept away to generate the windows in the data stream (i.e., simulating new unseen data with a similar distribution of the training). Drift is simulated by removing one of the class labels during training and presenting such examples later during the online phase.

For instance, use case 1.1 contains 120k samples in the training and 7.6k in the test balance distributed across the 4 labels. From the training and test, we removed all the samples related to the *Science/Tech* label that will be used to simulate drift (32k). From the training, we kept away 31k samples to generate the new data streams. As a result, 60k samples from labels *World*, *Business*, and *Sport* were used for training and to model the baseline. 5.7k samples from the test belonging to the same labels were used to test the model's performance and estimate the threshold. The portion of the dataset unused (31k) and the drifted samples are used to generate the data streams for the controlled drift experiments. The same approach applies to all the use cases. The *offline* phase (§2.1) is already executed for all those use cases.

Data stream generation. For each use case, the user can generate a data stream by setting two parameters: the *number of windows* and the *window size*. A data stream consisting of *number of windows* will be created, each composed of *window size* samples. Due to the dimensionality constraints, the stream is generated by random sampling with replacement. In the generation of the data stream, the user can simulate four types of scenarios by setting other parameters specific to each pattern, as shown in Figure 3:

- *No Drift*. None of the generated windows contain drifted samples. This pattern simulates the arrival of windows similar to the ones used during training.
- *Sudden Drift*. Concept drift occurs abruptly at a point in time. The user can specify the starting window from which drift will occur and the severity (i.e., the percentage of drifted samples in the windows).
- *Incremental Drift*. Concept drift occurs from a certain time but slowly, continuously, and incrementally. The user can specify the severity of the first drift occurrence and the increment step of each successive window.
- *Periodic Drift*. The old and new concepts reoccur repeatedly after some time. The user can specify the starting window from which the drift will occur, its duration, and its severity.

Once the use case selection and the data stream configuration are completed, the tool generates the data stream. Then, it starts the *online* phase by opening the drift detection monitor (§3.2).

3.2 Drift detection monitor

The tool processes each data window in real-time using the DriftLens methodology. The drift monitor page shows the distribution distances for the entire window (*per-batch*) and separately per label in two charts (*per-label*). The charts are dynamically

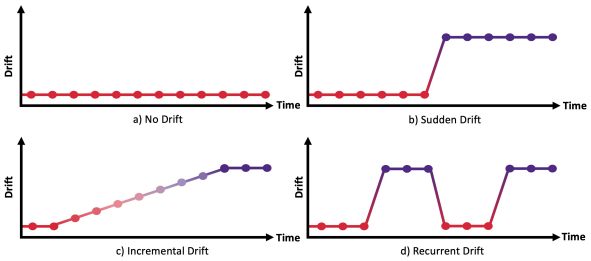

Figure 3: Drift patterns in the controlled experiments.

Figure 4: Drift detection monitor.

updated once DriftLens has processed each window until the end of the data stream. For each window, the tool executes steps ⑤, ⑥, ⑦, and ⑧ in Figure 1, and updates the charts accordingly.

Figure 4 shows an example of the drift monitor page for a data stream generated with a periodic drift pattern. The above chart shows the *per-batch* distribution distances for all the windows. Instead, the bottom one shows the distances separately for each label. The *x-axis* reports the timestamps and the window identifiers, while the distribution distances are in the *y-axis*. When drift is detected (i.e., the distribution distance in a given window is above the threshold), the area under the curve is filled, and a warning is displayed in the x-tick of the charts.

The user can exploit the monitor to understand: i) *when* drift occurs by looking at the windows in which drift was predicted; ii) *how* concept drift occurs in terms of severity of the drift (i.e., the greater the distance, the higher the severity) and patterns (i.e., if it is incremental, periodic, etc.); and iii) *where* by analyzing the labels the most affected by drift. In this case, the user can observe that drift occurs for the first time after 50 windows, with high severity, and then reoccurs intermittently with non-drifted windows. The label *World* is the most affected by drift, followed by *Business*. Instead, the label *Sports* is negligibly impacted.

3.3 Run drift detection on user-provided data

The second page enables users to experiment with drift detection on their data and models (Figure 5). To this end, users should provide the embedding and predicted labels[2] for the *baseline* and *threshold* datasets (e.g., training and test set). Those datasets are used to execute the *offline* phase of DriftLens.

Then, users should provide an order data stream and set the *window size* parameter. The data stream is split into windows,

[2]The correct dataset format is specified in the repository.

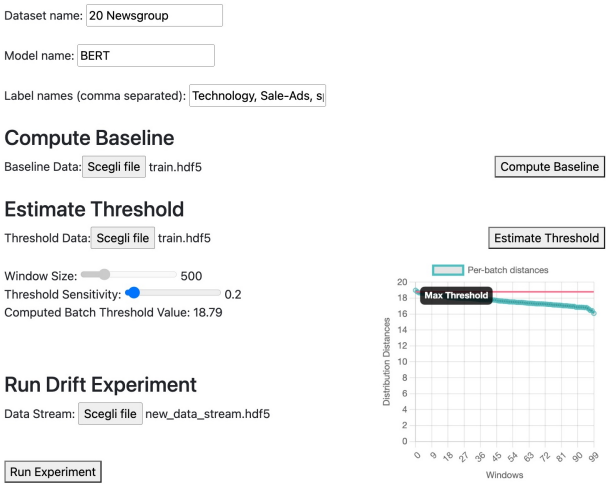

Figure 5: Drift detection on user-provided data.

maintaining the order of the data. Finally, after setting the threshold sensitivity, users can run *online* phase to perform drift detection and understand *if*, *where*, and *how* drift occurs on their data stream. The same drift monitor page is also open in this case (3.2). In this way, users can verify if their models and data are affected by drift.

4 CONCLUSION

This paper presents a tool based on the DriftLens methodology to detect concept drift in text and image classifiers. The tool enables users to i) experiment with pre-loaded use cases on controlled drift scenarios and ii) detect the presence of drift in their own data and models. The tool provides an effective real-time drift monitoring page that shows *when*, *how*, and *where* drift occurs in a data stream.

In future work, we would like to i) employ explainability techniques for deep models on unstructured data [20, 21] to interpret the reasons for change, and ii) address the adaptation problem by enabling the annotation of subsamples of the new data with human-in-the-loop and retraining the original model on the new concepts.

ACKNOWLEDGMENTS

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"); and the grant "National Centre for HPC, Big Data and Quantum Computing," CN000013 (approved under the M42C Call for Proposals - Investment 1.4 - Notice"National Centers" - D.D. No. 3138 of 16.12.2021, admitted for funding by MUR Decree No. 1031 of 17.06.2022).

We would like to thank Francesco Ventura and Paolo Bethaz for their valuable contributions and insightful discussions during the initial phase of this project.

REFERENCES

[1] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems* 245 (2022), 108632. https://doi.org/10.1016/j.knosys.2022.108632

[2] Li Bu, Cesare Alippi, and Dongbin Zhao. 2018. A pdf-Free Change Detection Test Based on Density Difference Estimation. *IEEE Transactions on Neural Networks and Learning Systems* 29, 2 (2018), 324–334. https://doi.org/10.1109/TNNLS.2016.2619909

[3] Tania Cerquitelli, Stefano Proto, Francesco Ventura, Daniele Apiletti, and Elena Baralis. 2019. Towards a real-time unsupervised estimation of predictive model degradation. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE 2019, Los Angeles, CA, USA, August 26, 2019*. ACM, 5:1–5:6. https://doi.org/10.1145/3350489.3350494

[4] Adam Coates, Andrew Ng, and Honglak Lee. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *Journal of Machine Learning Research - Proceedings Track* 15 (01 2011), 215–223.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Association for Computational Linguistics, Minneapolis, Minnesota. https://doi.org/10.18653/v1/N19-1423

[6] Gregory Ditzler and Robi Polikar. 2011. Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*. 41–48. https://doi.org/10.1109/CIDUE.2011.5948491

[7] Denis Moreira dos Reis, Peter Flach, Stan Matwin, and Gustavo Batista. 2016. Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1545–1554. https://doi.org/10.1145/2939672.2939836

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). https://arxiv.org/abs/2010.11929

[9] D.C Dowson and B.V Landau. 1982. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis* 12, 3 (1982), 450–455. https://doi.org/10.1016/0047-259X(82)90077-X

[10] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos. 2020. An overview of unsupervised drift detection methods. *WIREs Data Mining and Knowledge Discovery* 10, 6 (2020), e1381. https://doi.org/10.1002/widm.1381

[11] Salvatore Greco and Tania Cerquitelli. 2021. Drift Lens: Real-time unsupervised Concept Drift detection by evaluating per-label embedding distributions. In *2021 International Conference on Data Mining Workshops (ICDMW)*. 341–349. https://doi.org/10.1109/ICDMW53433.2021.00049

[12] Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

[13] Fabian Hinder, Valerie Vaquet, and Barbara Hammer. 2023. One or Two Things We know about Concept Drift – A Survey on Monitoring Evolving Environments. arXiv:cs.LG/2310.15826

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[15] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. 2019. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2019). https://doi.org/10.1109/TKDE.2018.2876857

[16] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. 2019. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/846c260d715e5b854ffad5f70a516c88-Paper.pdf

[17] Mohammad Rahimzadeh, Soroush Parvin, Elnaz Safi, and Mohammad Reza Mohammadi. 2021. Wise-SrNet: A Novel Architecture for Enhancing Image Classification by Learning Spatial Resolution of Feature Maps. *CoRR* abs/2104.12294 (2021). arXiv:2104.12294 https://arxiv.org/abs/2104.12294

[18] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:cs.CL/1910.01108

[19] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[20] Francesco Ventura, Salvatore Greco, Daniele Apiletti, and Tania Cerquitelli. 2022. Trusting deep learning natural-language models via local and global explanations. *Knowledge and Information Systems* (2022). https://doi.org/10.1007/s10115-022-01690-9

[21] Francesco Ventura, Salvatore Greco, Daniele Apiletti, and Tania Cerquitelli. 2023. Explaining deep convolutional models by measuring the influence of interpretable features in image classification. *Data Mining and Knowledge Discovery* (2023), 1–58.

[22] Francesco Ventura, Stefano Proto, Daniele Apiletti, Tania Cerquitelli, Simone Panicucci, Elena Baralis, Enrico Macii, and Alberto Macii. 2019. A New Unsupervised Predictive-Model Self-Assessment Approach That SCALEs. In *2019 IEEE International Congress on Big Data, Milan, Italy, July 8-13, 2019*. IEEE, 144–148. https://doi.org/10.1109/BIGDATACONGRESS.2019.00033

[23] Elias Werner, Nishant Kumar, Matthias Lieber, Sunna Torge, Stefan Gumhold, and Wolfgang E. Nagel. 2023. Examining Computational Performance of Unsupervised Concept Drift Detection: A Survey and Beyond. arXiv:cs.LG/2304.08319

[24] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS*.