Biomedical Informatics Insights

ORIGINAL RESEARCH

Binary Classifiers and Latent Sequence Models for Emotion Detection in Suicide Notes

Colin Cherry, Saif M. Mohammad and Berry de Bruijn

Institute for Information Technology, National Research Council Canada Ottawa, Ontario, Canada, K1A 0R6.
Corresponding author email: Colin.Cherry@nrc-cnrc.gc.ca

Abstract: This paper describes the National Research Council of Canada's submission to the 2011 i2b2 NLP challenge on the detection of emotions in suicide notes. In this task, each sentence of a suicide note is annotated with zero or more emotions, making it a multi-label sentence classification task. We employ two distinct large-margin models capable of handling multiple labels. The first uses one classifier per emotion, and is built to simplify label balance issues and to allow extremely fast development. This approach is very effective, scoring an F-measure of 55.22 and placing fourth in the competition, making it the best system that does not use web-derived statistics or re-annotated training data. Second, we present a latent sequence model, which learns to segment the sentence into a number of emotion regions. This model is intended to gracefully handle sentences that convey multiple thoughts and emotions. Preliminary work with the latent sequence model shows promise, resulting in comparable performance using fewer features.

Keywords: natural language processing, text analysis, emotion classification, suicide notes, support vector machines, latent variable modeling

Biomedical Informatics Insights 2012:5 (Suppl. 1) 147–154

doi: 10.4137/BII.S8933

This article is available from http://www.la-press.com.

Introduction

A suicide note is an important resource when attempting to assess a patient's risk of repeated suicide attempts.[1,2] Task 2 of the 2011 i2b2 NLP Challenge examines the task of automatically partitioning a note into regions of emotion, instruction, and information, in order to facilitate downstream processing and risk assessment.

The provided corpus of emotion-annotated suicide notes is unique in many ways.[3] The text is characterized by being emotionally-charged, personal, and raw. By definition, the writing is the product of a distressed mind, often containing spelling mistakes, asides, ramblings, and vague allusions. The text's unedited nature makes it difficult for basic NLP tools: optical character recognition, tokenization and sentence breaking errors abound. The provided emotion annotation also has some interesting properties. Annotation is given at the sentence level within complete documents, and each sentence can be assigned multiple labels. Where one would expect most of a suicide note to be rich with emotion, more than 73% of the training sentences are annotated with no label or with only *instructions* or *information*. This leaves very little data for the thirteen remaining labels, which include more conventional emotions such as *anger*, *fear* and *love*.

We experiment with two different large-margin learning algorithms, each using a similar, knowledge-light feature set. The first approach trains one classifier to detect each label, a strategy that has been applied successfully to other multi-label tasks.[4–6] The second approach attempts to segment a sentence into multiple emotive regions using a latent sequence model. This novel method looks to maintain the strengths of a Support Vector Machine (SVM) classifier, while enabling the model to gracefully handle run-on sentences. However, because so much of the data bears no emotion at all, the issue of class balance becomes a dominating factor. This gives the advantage to the one-classifier-per-label system, which uses well-understood binary classifiers and produces our strongest result. It scores 55.22 in micro-averaged F-measure, finishing in fourth place. The latent sequence model scores 54.63 in post-competition analysis, and produces useful phrase-level emotion annotations, indicating that it warrants further exploration.

Related Work

Over the last decade, there has been considerable work in sentiment analysis, especially in determining whether a term, sentence, or passage has a positive or negative polarity.[7] Work on classifying terms, sentences and documents according to the emotions they express, such as anger, joy, sadness, fear, surprise, and disgust is relatively more recent, and it is now starting to attract significant attention.[3,8–10]

Emotion analysis can be applied to all kinds of text, but certain domains tend to have more overt expressions of emotions than others. Computational methods have not been applied in any significant degree to suicide notes, largely because of a paucity of digitized data. Work from the University of Cincinnati on distinguishing genuine suicide notes from either elicited notes or newsgroup articles is a notable exception.[1,2] Neviarouskaya et al,[9] Genereux and Evans,[11] and Mihalcea and Liu[12] analyzed web-logs. Alm et al,[13] Francisco and Gervá,[14] and Mohammad[15] worked on fairy tales. Zhe and Boucouvalas,[16] Holzman and Pottenger,[17] and Ma et al[18] annotated chat messages for emotions. Liu et al[19] and Mohammad and Yang[20] worked on email data. Much of this work focuses on the six emotions studied by Ekman[21]: joy, sadness, anger, fear, disgust, and surprise. Joy, sadness, anger, and fear are among the fifteen used in i2b2's suicide notes task.

Automatic systems for analyzing emotional content of text follow many different approaches: a number of these systems look for specific emotion denoting words,[22] some determine the tendency of terms to co-occur with seed words whose emotions are known,[23] some use hand-coded rules,[9] and some use machine learning.[8,13] Machine learning is also a dominant approach for general text classification.[4] Tsoumakas and Katakis[6] provide a good survey of machine-learning approaches for multi-label classification problems, including text classification.

One Classifier Per Label

The provided training data consists of sentences annotated with zero or more labels, making it a multi-label classification problem. One popular framework for multi-label classification builds a separate binary classifier to detect the presence or absence of each label.[4,5] The classifiers in our One-Classifier-Per-Label (1CPL)

system are linear SVMs, each trained for one of the emotions by using as positive training material those sentences that bear the corresponding label, and as negative training material those sentences that do not. To counter the strong bias effects in this data, we adjust the weight of each class so that negative examples have less impact on learning.

Model

We view our training data as a set of training pairs $[\bar{x}_i, Y_i]_{i=1}^n$, where $\bar{x}_i$ is a feature vector representing an input sentence, and Y_i is the correct set of labels assigned to that sentence. Note that each sentence is labeled independently of the other sentences in a document. There is a finite set of possible labels $\mathcal{Y}$, such that $\forall i : Y_i \subseteq \mathcal{Y}$. In the case of the i2b2 emotion data, $\mathcal{Y} = \{abuse, anger, \ldots\}$ and $|\mathcal{Y}| = 15$.

The 1CPL model builds a binary classifier for each of the $|\mathcal{Y}|$ labels, creating one classifier to detect the presence or absence of *abuse*, and another for *anger*, and so on. For each emotion $e \in \mathcal{Y}$, we construct a training set $D^e = [\bar{x}_i, y_i^e]_{i=1}^n$ such that: $y_i^e = +1$ if $e \in Y_i$ and $y_i^e = -1$ otherwise.

D^e is divided into positive examples D_+^e and negative examples D_-^e based on y_i^e. For each $e \in \mathcal{Y}$, a linear-kernel SVM learns a weight vector $\bar{w}^e$ by optimizing a standard (class-weighted) objective:

$$\forall e \in \mathcal{Y}: \bar{w}^e = \arg\min_{\bar{w}} \frac{1}{2} \| \bar{w} \|^2 + C_* \left(C_+^e \sum_{[\bar{x}_i, y_i^e] \in D_+^e} l(\bar{w}, y_i^e, \bar{x}_i) + C_-^e \sum_{[\bar{x}_i, y_i^e] \in D_-^e} l(\bar{w}, y_i^e, \bar{x}_i) \right) \quad (1)$$

where $l(\bar{w}, y, \bar{x})$ is the binary hinge loss. Note that we have $1 + 2|\mathcal{Y}|$ hyper-parameters: C_*, plus two weights C_+^e and C_-^e for each label. C_* is shared by all classifiers and controls the over-all emphasis on regularization. The label-specific weights control for class imbalance. Even the most frequent label (*instructions*) is infrequent, occurring in fewer than 18% of training sentences; therefore, a standard SVM will learn a strong bias for the negative class. Since our goal is to achieve high F-measure, which balances precision and recall equally, we must correct for this bias. To do so, C_+ and C_- are set automatically to make positive and negative classes contribute equally to the total hinge loss: $C_+^e = 1/|D_+^e|$ and $C_-^e = 1/|D_-^e|$. C_* is tuned by grid-search using 10-fold cross-validation. Given models $\bar{w}^e$ for each $e \in \mathcal{Y}$ and a novel sentence $\bar{x}$, the retrieval of a label set $Y(\bar{x})$ is straight-forward: $Y(\bar{x}) = \{e \mid \bar{w}^e \cdot \bar{x} > 0\}$.

We train each model using an in-house SVM that is similar to LIBLINEAR.[24] Training is very fast, allowing us to test all 15 classifiers over 10 folds in less than a minute on a single processor. Our selection criterion for both feature design and hyper-parameter tuning is micro-averaged F-measure, as measured on the complete training set after 10-fold cross-validation. We now describe the feature templates used in our system, grouped into thematic categories:

Base

Our base feature set consists of a bias feature that is always on, along with a bag of lowercased word unigrams and bigrams. This base vector is normalized so that $\| \bar{x} \| = 1$. As other feature templates are added, they use the value corresponding to 1 in this normalization to scale their counts.

Sentence

This template summarizes the sentence in broad terms, using features like its length in tokens. This includes features that check for the presence of manually-designed word classes, reporting whether the sentence contains any capitalized words, only capitalized words, any anonymized names (John, Jane), any future-tense verbs, and so on.

Thesaurus

We included two sources of hand-crafted word clusters. The first counts the words matching each category from the freely available *Roget's Thesaurus*.[1] The second uses the thesaurus to seed a much smaller manually-crafted list of near-synonyms for the 15 competition labels (16 after *happiness_peacefulness* is split into its component words). For example, the *hopefulness* list contains "hopefulness", "hopefully" and "hope" as well as "optimistic", "confident"

[1] *Roget's Thesaurus*: www.gutenberg.org/ebooks/10681.

and "faith". A feature is generated to count matches to each near-synonym list.

Character

This template returns the set of cased character 4-grams found in the sentence. These features are intended as a surrogate for stemming and spelling correction, but they also reintroduce case to the system, which otherwise uses only lowercased unigrams and bigrams.

Document

Four document features describe the document's length and its general upper/lower-case patterns. Document features are identical across each sentence in a document.

Experiments

Table 1 reports the results of our 1CPL submissions to the competition in terms of micro-averaged precision, recall and F-measure. System 1 was intended to test our pipeline and output format, and was submitted before tuning was complete. System 2 uses all of the features listed above, with C_* carefully tuned through cross-validation. We use the remainder of this section to analyze System 2's performance in greater detail.

Ablation

Table 2 shows the results of our ablation experiments. With each modification, we re-optimized C* in 10-fold cross-validation and then froze it for the corresponding test set run. First, we removed the class balance parameters C_+^e and C_-^e from our base system, revealing a dramatic drop in recall. We then added each template to the base system alone. 10-fold cross-validation on the training set indicates that three out of four templates are helpful. However, on the test set, only the character *n*-grams improve upon our baseline performance. In fact, using the *n*-grams alone exceeds our complete system's performance, producing the strongest reported result that we know of that uses no external resources (no thesauri, auxiliary corpora, web data, or manual re-annotations).

Table 1. Test-set performance of our 1CPL systems.

Description	Precision	Recall	F1
System 1: 1CPL test	48.71	56.45	52.29
System 2: 1CPL full	**55.72**	**54.72**	**55.22**

Per-label performance

Table 3 shows our performance as decomposed over each label. With the exception of *thankfulness*, all of our high-performing labels correspond to high-frequency labels. Our SVM classifier excels when given sufficient training data; furthermore, by tuning for micro-averaged F-measure during development, we necessarily place emphasis on the high-frequency labels. Both *thankfulness* and *love* appear to be relatively easy for our system to handle, likely due to these emotions having a relatively small set of short and common lexical indicators.

Label confusion

We also examined those cases where both the model and the gold-standard agreed a sentence should be labeled, but they did not agree on which labels. We found only three sources of substantial confusion in the test set: *information* and *instructions* were confused 26 times, *guilt* and *sorrow* were confused 10 times, and *guilt* and *hopelessness* were confused 9 times. The vast majority of our errors consisted of either assigning a label to a sentence that should be left unlabeled, or erroneously leaving a sentence unlabeled.

Discussion

The 1CPL model is very fast to train, and our automatically determined class weights C_+^e and C_-^e address the issue of class imbalance without introducing more hyper-parameters to be tuned. This proved to be very useful; most other learning approaches we attempted quickly became bogged down in an extensive hyper-parameter search.

Unfortunately, 1CPL cannot reason about multiple label assignments simultaneously. No one classifier knows what other labels will be assigned, nor does it know how many labels will be assigned. This means that we cannot model when a sentence should be given no labels at all, nor can we model the fact that longer sentences tend to accept more labels. Our next approach attempts to address some of these weaknesses.

Latent Sequence Model

On many occasions, when a sentence expresses multiple emotions, it can be segmented into different parts, each devoted to one emotion. Take for example,

Table 2. Ablation experiments for our 1CPL model. Results exceeding the baseline are in bold. Cross-validation results are micro-averaged scores measured on the entire training set (4241 sentences, 600 documents, 2522 gold labels). Test results are on the official test set (1883 sentences, 300 documents, 1272 gold labels).

System version	Cross-validation			Test		
	Precision	Recall	F1	Precision	Recall	F1
Base	54.04	51.94	52.97	55.26	52.83	54.02
$-C_+^e$ and C_-^e	66.74	38.18	48.58	70.62	40.25	51.28
+ sentence	52.40	53.25	52.82	52.51	54.25	53.36
+ thesaurus	53.11	54.20	**53.65**	51.94	54.80	53.33
+ character	55.51	52.50	**53.96**	57.86	53.54	**55.61**
+ document	53.09	54.92	**53.99**	53.03	55.03	54.01
+ all	55.29	54.12	**54.70**	55.72	54.72	**55.22**

the following sentence, which was tagged with *{hopelessness*, *love}*:

> Dear Jane I love you but I am so weak.

It is easy to see that this could be segmented into emotive regions. We represent these regions using a tag sequence, with tags drawn from $\mathcal{Y} \cup \{O\}$, where O is an **outside tag**, which represents regions without emotion:

O	love	*O*	hopelessness	*O*
Dear Jane	I love you	but	I am so weak	.

Inspired by latent subjective regions in movie reviews,[25] our latent sequence model assumes that each training sentence has a corresponding latent (or hidden) tag sequence that was omitted from the training data. Our latent sequence model attempts to recover these hidden sequences using only the standard training data, requiring no further annotation.

By learning a tagger instead of a classifier, we hope to achieve three goals. First, we hope to learn more precisely from sentences that have multiple labels: a *love* region will not contribute features to the *hopelessness* model. Second, longer sentences will have more opportunities to express multiple emotions. Finally, modeling emotion sequences within a sentence should help recover from errors in the provided sentence segmentation. We train our latent sequence model by iteratively improving a discriminative tagger. Beginning with a weak model, we find the best tag sequence for each training sentence that uses only and all the gold-standard labels for that sentence. We then use these tag sequences as training data to retrain our discriminative tagger. The process iterates until performance stops improving.

Table 3. Per-label performance of our best 1CPL system as measured on the test set.

Label	# Right	# Guessed	# Gold	Precision	Recall	F1
Abuse	0	0	5	–	0.00	–
Anger	1	4	26	25.00	3.85	6.67
Blame	2	19	45	10.53	4.44	6.25
Fear	0	1	13	0.00	0.00	0.00
Forgiveness	0	0	8	–	0.00	–
Guilt	51	112	117	45.54	43.59	44.54
Happiness_peacefulness	1	1	16	100.00	6.25	11.76
Hopefulness	3	7	38	42.86	7.89	13.33
Hopelessness	149	291	229	51.20	65.07	57.31
Information	45	125	104	36.00	43.27	39.30
Instructions	269	416	382	64.66	70.42	67.42
Love	145	215	201	67.44	72.14	69.71
Pride	0	0	9	–	0.00	–
Sorrow	0	8	34	0.00	0.00	0.00
Thankfulness	30	50	45	60.00	66.67	63.16
All labels	696	1249	1272	55.72	54.72	55.22

Model

We view our training data as a set of training pairs $[x_i, Y_i]_{i=1}^n$ where x_i is a tokenized input sentence, and Y_i is a gold-standard set of labels. We represent a latent tag sequence with the variable h. To bridge emotive tag sequences h and emotion sets Y, we make use of a function $set(h)$:

$$set(h) = \{e \mid e \neq O \wedge e \text{ is used as a tag in } h\}$$

For example, for h = "*O love O hopelessness O*", $set(h) = \{love, hopelessness\}$. To go from a set Y to a sequence h, we need a model $\bar{w}$ and a function *tags*, which finds the best sequence that uses only and all the emotions from Y:

$$tags(\bar{w}, x, Y) = \arg\max_{h:set(h)=Y} \bar{w} \cdot \bar{f}(x, h) \quad (2)$$

Here $\bar{f}$ is a feature function that maps x, h pairs to feature vectors that decompose into dynamic programming, as in perceptron tagging.[26] The argmax over tag sequences can be computed using a semi-Markov tagger.[27] The tagger can be constrained to match Y by using a standard k-best list extraction algorithm to pop complete hypotheses from the dynamic programming chart until an h satisfying $set(h) = Y$ is found.

We now find ourselves in a circular situation. If we had a tagging model $\bar{w}$, we could use Equation 2 to find tag sequences h_i for each $[x_i, Y_i]$ in the training data. Meanwhile, the resulting $[x_i, h_i]$ pairs could provide training data to learn a tagging model $\bar{w}$. This circularity should be familiar to anyone who has worked with the EM algorithm,[28] and to address it, we will use a latent SVM,[29] a discriminative, large-margin analogue to EM.

The latent SVM alternates between building training sequences and learning sequence models. Given an initial $\bar{w}$, the following two steps repeat m times (we initialize $\bar{w}$ by adapting weights from the 1CPL system).

1. For each training point, transform label sets Y_i into sequences h_i using $h_i = tags(\bar{w}, x_i, Y_i)$
2. Learn a new $\bar{w}$ by training a structured SVM to predict h_i from x_i. We use the Pegasos primal gradient optimization algorithm.[30]

The structured SVM optimizes its weights according to the following objective:

$$\bar{w} = \arg\min_{\bar{w}} \frac{\lambda}{2} \| \bar{w} \|^2 + \sum_i l(\bar{w}, x_i, h_i) \quad (3)$$

where λ is a regularizing hyper-parameter and $l()$ generalizes the hinge loss to structured outputs:

$$l(\bar{w}, x_i, h_i) = \max_h [\Delta(h_i, h) + \bar{w} \cdot \bar{f}(x_i, h)] - \bar{w} \cdot \bar{f}(x_i, h_i) \quad (4)$$

and $\Delta(h_i, h)$ is a cost function that defines how incorrect h is with respect to h_i. By minimizing Equation 3, we encourage each incorrect h to score lower than the oracle h_i with a margin of $\Delta(h_i, h)$. Careful design of Δ is essential to creating an efficient latent SVM with strong performance. In our case, we define a *bag*( ) function analogous to *set*( ), and use that to define a weighted hamming distance over bags:

$$\Delta(h_i, h) = \delta_o \mid set(h_i) \mid - \sum_{e \in bag(h) \wedge e \in set(h_i)} \delta_o + \sum_{e \in bag(h) \wedge e \notin set(h_i)} \delta_c \quad (5)$$

where δ_c and δ_o are hyper-parameters corresponding to costs for errors of commission and omission respectively. These values adjust the model's precision-recall trade-off, and are determined empirically using cross-validation. We use bags as an approximation to the sets we truly care about because the bag-based cost function factors nicely into dynamic programming, allowing us to calculate the max in Equation 4, which is necessary for Pegasos training.

The latent sequence model is substantially slower than 1CPL, but still quite manageable. We can train and test one inner loop in about 12 minutes, meaning all 10 folds can be tested in 2 hours. The system rarely requires more than $m = 3$ loops to achieve good results. However, in the context of a competition, this speed hampers rapid tuning and feature development. Hence our submitted system is only roughly tuned and uses only straight-forward features.

We adapt the baseline and thesaurus features from 1CPL to our latent sequence model. Recall that a tag sequence h is a sequence of emotion-tagged substrings, such that no two substrings overlap. Each substring generates features for the unigrams and bigrams ending in that substring, as well as any thesaurus matches. As is standard in discriminative tagging, the current tag (drawn from $\mathcal{Y} \cup \{O\}$) is appended to each feature.

Experiments

Table 4 shows the performance of two latent sequence models on the competition test set. Our official System 3 used $\lambda = 0.01$, $\delta_c = 1/15$, $\delta_o = 5/15$, and $m = 1$, which were the best settings we could find in time for the competition. We have also included System *, which corresponds to better parameters that were found using 10-fold cross-validation after the competition closed. It changes λ to 0.03 and m to 2. As one can see, the latent system does not outperform 1CPL, despite our intuitions regarding the utility of latent emotive regions. However, we do feel it shows promise, as it does not yet use all of the features of 1CPL, and the over-all system design may have room for further refinement. We intend to continue improving its performance. It is interesting to look at some of the tagger's outputs; here we draw examples from the training data as tagged by cross-validation. Often, the system behaves exactly as intended:

$Y_i = \{guilt, hopelessness\}$

hopelessness	guilt
I cant get well so whats the use living	God forgive me

The current version is also prone to over-segmentation, and does not always split lines at reasonable positions:

$Y_i = \{guilt\}$

instructions	guilt	O
Tell him to	forgive me if I ever treated	him bad.

Table 4. Test-set performance of our latent sequence models.

Description	Precision	Recall	F1
System 3: Latent (submitted)	48.15	58.18	52.69
System *: Latent (post-eval)	54.78	54.48	54.63

This indicates that the system could benefit from more feature that help it detect good split points. There are also cases where our one-emotion-per-region assumption breaks down, with two emotions existing simultaneously. Here, two emotions label one clause, and the system catches neither:

$Y_i = \{anger, blame\}$

O
You think you are so wonderful.

Like the 1CPL system, the majority of our errors correspond to problems with empty sentences: either leaving emotion-bearing sentences empty, or erroneously assigning an emotion to an empty sentence.

Conclusion

We have analyzed two machine learning approaches for multi-label text classification in the context of emotion annotation for suicide notes. Our one-classifier-per-label system provides a clean solution for class imbalance, allowing us to focus on feature design and hyper-parameter tuning. It scores 55.22 F1, placing fourth in the competition, without the use of web-derived statistics or re-annotated training data. Beyond basic word unigrams and bigrams, its most important features are character 4-grams.

We have also presented preliminary work on a promising alternative, which handles multiple labels by segmenting the text into latent emotive regions. This gracefully handles run-on sentences, and produces easily-interpretable system output. This approach needs more work to match its more mature competitors, but still scores favorably with respect to the other systems, receiving an F1 of 54.63 in post-competition analysis.

Disclosures

Author(s) have provided signed confirmations to the publisher of their compliance with all applicable legal and ethical obligations in respect to declaration of conflicts of interest, funding, authorship and contributorship, and compliance with ethical requirements in respect to treatment of human and animal test subjects. If this article contains identifiable human subject(s) author(s) were required to supply signed patient consent prior to publication. Author(s) have