Feature Design for Transfer Learning

Mark Dredze MDREDZE@CIS.UPENN.EDU
John Blitzer BLITZER@CIS.UPENN.EDU
Koby Crammer CRAMMER@CIS.UPENN.EDU
Fernando Pereira PEREIRA@CIS.UPENN.EDU
Department of Computer and Information Science, University of Pennsylvania

Abstract

Discriminative learning methods for classification perform well when training and test data are drawn from the same distribution and labeled using the same function. However, often we have labeled data for a task related to the target task but not for the target task itself. Under what conditions does a good classifier for the related task transfer to the target task? Feature representations that capture uniformities between the two tasks are a crucial factor in the success of transfer. We formalize this intuition theoretically with a generalization bound for transfer, and demonstrate it empirically with a particular kind of representation, user-centric features, that helps transfer an email reply predictor between different email users.

1. Transfer Learning

Human beings can naturally transfer knowledge about a *source* task to perform better on a closely related *target* task. For example a veteran clerk beginning a job at a new store has experience that allows her to operate at a high level of proficiency from the start. An experienced secretary will easily adapt to scheduling appointments in a computer science department, even if his former job was in the psychology department. This is the basic principle behind transfer learning: a system trained on a source task should achieve good performance on related target tasks faster than untrained systems.

Discriminative learning methods for classification are based on the assumption that training and test data are drawn from the same distribution. This assumption underlies both theoretical estimates of generalization error and the many experimental evaluations of learning methods. However, often we have little or no training data for a target task, but we have plentiful training data for a related source task. In those situations, we would like to *transfer* the classifier learned for the source task to the target task.

In this work we focus on transfer via a common feature representation for tasks. By designing a good representation, the two tasks are mapped together so that a learner for one task automatically performs well on another, without meta-learning or additional training data for the new task. We give theoretical results showing that representation is crucial to transfer performance. We further show significant improvements on the real-world problem of email reply prediction, where we have only one source task and no training data for the target task.

In email reply prediction, the goal of the learner is to predict if an email user will reply to a given message. The task is user-dependent in two ways. First, each user receives a different message stream. Second, each user has his own rule for replying to messages. We transfer a classifier between two users through the design of *user-centric* (or deictic) features. We call these features user-centric because each task involves a different email recipient, and the features capture relations between messages and recipients, rather than intrinsic properties of the messages.

Transfer learning is an important problem in machine learning, and it has been studied in many different areas. Some theoretical analyses of transfer that are closely related to our work are Baxter (1995) , who introduces the process of learning a common representation or "environment" from which discriminative hypotheses may be drawn. Ben-David (2003) described task-relatedness in terms of mappings from a domain of learning $\mathcal{X}$ to itself. Crammer et al. (2005) investigate a setting where data drawn from a fixed unknown distribution is partitioned into subsets labeled by similar but not identical concepts.

These works assume the existence of several related source tasks, all of which have training data. We do not, and it is important to observe that our method gives significant gains even with a single source task. Sutton and McCallum (2005) also use training data from only one source task, but they assume training data in the target task. One of our

Preliminary work. Appeared in the North East Student Colloquim on Artificial Intelligence.

most crucial contributions is that our work is effective with no training data from the target task.

The remainder of this paper is structured as follows: Section 2 describes in greater detail our problem setting. Section 3 gives a precise definition of transfer representations and an error bound that demonstrates the importance appropriate representations. Section 4 introduces email reply prediction, our experimental task that requires a transferable model. Sections 5-6 show empirically how *user-centric* features, designed for transfer among email users, significantly improve this task. We conclude with a discussion of future research directions.

2. Problem Setup

In the standard classification setting for complex instances like email messages, the instance is first transformed into a vector representation whose components are called features. Obviously, the representation should capture those properties of instances that are predictive of the correct classification. For transfer learning, however, predictiveness is not enough. We need feature representations that also support successful transfer of the classifier learned for the source task to a target task.

We return to our task of email reply prediction. A user's unique message stream corresponds formally to a distinguished distribution over his inbox. His rule for replying to messages corresponds formally to a classification function. Consider two email users, Jarvis[1] and Merrick.[2] Assume that Jarvis likes to ski and Merrick likes to surf. If both receive messages related to their hobbies, clearly their message distributions are different. For instance, we expect Merrick to receive messages containing words like *wave* and *surfboard*, and Jarvis to receive messages containing words like *snow* and *skis*. Even if we could map between the dictionaries related to each of the hobbies, the way the two users reply to messages about their hobbies may be different. Jarvis might reply to most incoming messages about his hobby, while Merrick might ignore most messages about his.

Our task, then, is to transfer an email reply predictor trained on Jarvis's data to Merrick. Our *user-centric* features attempt to make Jarvis's and Merrick's distribution over emails similar by mapping features unique to each user onto features that are common for both users. Because of their design a classifier trained on a user-centric representation to minimize the empirical risk of Jarvis's reply prediction task should also perform well on Merrick.

[1] A conqueror.

[2] Ruler of the sea.

3. Theoretical Analysis

We now give a precise definition of transfer and analyze the generalization error of a classifier trained in the source task when applied to the target task. The generalization error bound will depend on properties of the feature representation as we suggested in the preceding informal discussion.

Let $\mathcal{X}$ be an instance set (possible messages, for example), $\mathcal{Z}$ be a feature space ($\mathbb{R}^d$ is a typical choice), and $\mathcal{Y} = \{0, 1\}$ the label set for binary classification (reply to this message or not, for example).

A learning problem is specified by two parameters: a distribution $\mathcal{D}$ over $\mathcal{X}$ and a (stochastic) target function $f : \mathcal{X} \to [0, 1]$. The value of $f(x)$ corresponds to the probability that the label of x is 1. Let $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ be a fixed representation function. The representation $\mathcal{R}$ induces a distribution over $\mathcal{Z}$ and a (stochastic) target function from $\mathcal{Z}$ to $\mathcal{Y}$ as follows:

$$\begin{aligned} \Pr_{\tilde{\mathcal{D}}}[A] &\stackrel{\text{def}}{=} \Pr_{\mathcal{D}}\left[\mathcal{R}^{-1}(A)\right] \\ \tilde{f}(z) &\stackrel{\text{def}}{=} \mathrm{E}_{\mathcal{D}}\left[f(x)|\mathcal{R}(x) = z\right] \end{aligned}$$

for any $A \subseteq Z$ such that $\mathcal{R}^{-1}(A)$ is D-measurable. In words, the probability of an event A under $\tilde{\mathcal{D}}$ is the probability of the inverse image of A under $\mathcal{R}$ according to $\mathcal{D}$, and the probability that the label of z is 1 according to $\tilde{f}$ is the mean of probabilities of instances x that z represents. Note that $\tilde{f}$ might be a stochastic function even if f is deterministic, because the function $\mathcal{R}$ may map two instances with different f-labels to the same feature representation.

In summary, our learning setting is defined by a fixed but unknown $\mathcal{D}$ and f, a fixed, known $\mathcal{R}$, and a hypothesis space $\mathcal{H} = \{h : \mathcal{Z} \to \mathcal{Y}\}$ of deterministic hypotheses to be used to approximate the function f to be learned.

We assume two email users (tasks), Jarvis and Merrick. We denote by $\mathcal{D}_J$ the distribution of received messages for Jarvis, $\tilde{\mathcal{D}}_J$ the induced distribution over the feature space $\mathcal{Z}$, and by f_J the rule Jarvis uses to reply to messages. We use parallel notations for Merrick.

We denote the probability according to the distribution $\mathcal{D}_J$, that a function h disagrees with f_J by

$$\begin{aligned} \epsilon_J^J(h) &= \mathrm{E}_{z\sim\tilde{\mathcal{D}}_J}\left[\mathrm{E}_{y\sim\tilde{f}_J(z)}\left[y \neq h(z)\right]\right] \\ &= \mathrm{E}_{z\sim\tilde{\mathcal{D}}_J}\left|\tilde{f}_J(z) - h(z)\right|. \end{aligned}$$

Assume that we have m_J pairs of messages and responses (z_i, y_i). The messages were drawn iid according to $\mathcal{D}_J$ from $\mathcal{X}$ and then represented using $\mathcal{R}$ and labeled iid according to f_J. We denote the frequency of examples in this set for which a function h does not agree with the target

label by

$$\hat{\epsilon}_J^J(h) = \frac{1}{m_J} \sum_i I_{y_i \neq h(z_i)} \ .$$

We define similar notation for all possible combinations of distributions ($\tilde{\mathcal{D}}_J$ or $\tilde{\mathcal{D}}_M$), target functions ($\tilde{f}_J$ or $\tilde{f}_M$), and true quantity or its estimate. For example, $\epsilon_M^J(h)$ is the error a function h suffers with respect to $\tilde{f}_J$ where the instances were drawn according to $\tilde{\mathcal{D}}_M$. Below, we use the following definition from Freund et al. (1997):

Definition 1 *A distribution $\mathcal{D}$ is within $\lambda \geq 1$ of a distribution $\mathcal{D}'$ if for every measurable set A, we have*

$$1/\lambda \leq \frac{\Pr_{\mathcal{D}}[A]}{\Pr_{\mathcal{D}'}[A]} \leq \lambda \ .$$

We now state the transfer learning problem. We first fix the representation function $\mathcal{R}$ and the hypothesis space $\mathcal{H}$. Next, we receive a set of m_J labeled examples and choose the empirical-risk minimizer

$$\hat{h}_J = \arg\min_h \hat{\epsilon}_J^J(h) \ .$$

We measure how well this hypothesis performs on Merrick's messages by bounding the true error of this hypothesis on Merrick's task relative to the *best* function Merrick can use. The bounds are of the form

$$\epsilon_M^M(\hat{h}_J) \leq a \ \min_h \epsilon_M^M(h) + b \ ,$$

comparing the performance of our chosen hypothesis with the best possible hypothesis using a multiplicative factor a and an additive factor b. The theorem below (its proof technique uses standard generalization analysis (Anthony & Bartlet, 1999) in addition to the previous definition. The proof is omitted for lack of space) summarizes our main theoretical result:

Theorem 1 *Let $\mathcal{R}$ be a fixed representation function from the instance space $\mathcal{X}$ to the instance space $\mathcal{Z}$ and let $\mathcal{H} = \{h : \mathcal{Z} \to \mathcal{Y}\}$ be a hypothesis space of VC-dimension d. Assume that $\hat{h}_J$ is the empirical risk minimizer $\hat{h}_J = \arg\min_h \hat{\epsilon}_J^J(h)$. Assume that $\tilde{\mathcal{D}}_J$ and $\tilde{\mathcal{D}}_M$ are within λ of each other. Then, with probability of at least $1-\delta$ the following bound holds:*

$$\begin{aligned}
\epsilon_M^M(\hat{h}_J) \quad \leq \quad & \min_h \epsilon_M^M(h) \\
& + 2\sqrt{\frac{d \log\left(\frac{2em_J}{d}\right) + \log\left(\frac{8}{\delta}\right)}{8m_J}} \\
& + 2 \min\left\{ \mathrm{E}_{z\sim\tilde{\mathcal{D}}_J}\left|\tilde{f}_J(z) - \tilde{f}_M(z)\right|, \right. \\
& \qquad \left. \mathrm{E}_{z\sim\tilde{\mathcal{D}}_M}\left|\tilde{f}_J(z) - \tilde{f}_M(z)\right| \right\} \\
& + 2(\lambda - 1) \ .
\end{aligned}$$

The representation function $\mathcal{R}$ affects the first, third and fourth terms of the bound. The second term arises from using a finite sample to choose the hypothesis. The first term

$$\min_h \epsilon_M^M(h) \ ,$$

is the performance of the best possible hypothesis. In the context of our example, this is the best reply prediction function for persons who are interested in surfing ($\tilde{f}_M$) evaluated on messages about surfing ($\tilde{\mathcal{D}}_M$). Note that if f is not a stochastic function, then one can choose $\mathcal{R}$ such that the performance of the best hypothesis would be the worst possible. If, for example, $\mathcal{R}$ maps $\mathcal{X}$ to a single point $z_0 \in \mathcal{Z}$, then $\tilde{f}(z_0)$ is just the probability that $f(x) = 1$. In this case the best classifier would suffer a loss of

$$\min\left\{\Pr_{\mathcal{D}_M}[f_M(x) = 1] \ , \ \Pr_{\mathcal{D}_M}[f_M(x) = 0]\right\} \ .$$

On the other hand, if the function $\mathcal{R}$ induces a deterministic function $\tilde{f}$ then the performance of the best hypothesis depends solely on the richness of the fixed hypothesis class $\mathcal{H}$.

The third term compares the performance of the two induced functions $\tilde{f}_J$ and $\tilde{f}_M$ using a single distribution. If the two functions are similar then the bound is small. Note again that mapping $\mathcal{X}$ into a single point will set this term to zero. As mentioned above, we assume that this term is indeed small, that is, the two users' reply functions have similar characteristics. For example, they will not reply to messages about places to enjoy their hobby, but they will to messages about new gear for their hobby.

Finally, the last term $\lambda - 1$, reflects the similarity between the two distributions $\tilde{\mathcal{D}}_J$ and $\tilde{\mathcal{D}}_M$ over the joint feature space. The more similar these two distributions, the smaller the bound. Mapping all points in $\mathcal{X}$ to a single point z_0 will set this term to zero. In terms of our running example, a good mapping would map *Whistler* and *Waikiki* into a token representing a location where the hobby may take place, and the words *ski* and *surfboard* into a single token representing hobby gear. Using such mapping we can transfer the essence of messages about one hobby to another.

To summarize, the bound reflects a trade-off in requirements for the representation function $\mathcal{R}$. On the one hand, we like simple representations, which will make the two distributions and target functions similar. On the other hand, representations that are too simple may lead to a loss of crucial information. A good representation function will balance between these two opposing requirements in two ways: we will be able to learn a good hypothesis for one of the tasks and we will be able to use this hypothesis successfully for another task.

4. Reply Prediction for Email Management

Email reply predicition is the task of automatically deciding whether or not a received email requires a reply. This is an ideal problem on which to demonstrate the theory of the previous section. As we mentioned in section 2, the terms users use to describe hobbies or jobs and the people they communicate with can differ significantly from user to user. Furthermore, users are notoriously intolerant of bad classification systems. For reply prediction, many users would rather not have any system than spend time and energy to train a completely new system for themselves from scratch.

4.1. Related Work on Email Management

Email has evolved to encompass a plethora of work-related activities. Whittaker and Sidner (1996) analyzed the use of email to perform task management, personal archiving, and asynchronous communication, and referred to the three as "email overload". They concluded that users perform a large variety of work-related tasks with email, and thus become overwhelmed by the amount of information in their mailboxes. A quotation from interviews conducted by Whittaker and Sidner (1996) characterizes some frustrations:

"Waiting to hear back from another ... employee can mean delays in accomplishing a particular task, which can ... have significant impact on our overall operations. ... it can be critical or just frustrating."

"One of my pet-peeves is when someone does not get back to me, but I am one of the worst offenders. I get so many emails ... that I cannot keep up."

In our experiments, we address the issue of waiting to hear back from others by learning to predict which messages need replies. Our system identifies and labels incoming messages that require a reply. Typically, a user must deal with a flood of new messages and decide which ones to read based on the subject and sender. The goal of our system is to empower the user to make a more informed decision about how to manage email by providing additional information.

Others have proposed solutions to the email overload problem. Dabbish et al. (2005) presents an email user survey highlighting factors that influence actions on messages, such as when to reply. Other systems approach this problem by providing social context for messages (Neustaedter et al., 2005). Remail (Boone, 1998) is an intelligent email agent that provides thread-arcs, fast search, and numerous other features to facilitate email management. Finally, a system which learned a (non-spam) email classifier is the "speech acts" system of Cohen et al. (2004). While this was not directly related to reply prediction, many of the speech acts they identified were quite similar.

While reply prediction, like spam detection, is a binary classification problem, they are quite different. Nearly all agree on what is spam and thus it can be aggregated to obtain a large pool of (positive) training examples. By contrast, legitimate emails sent to a group may require only one person's reply. Additionally, keywords are less useful in reply prediction, while social network factors are very good predictors.

4.2. Reply Prediction System

We implemented reply prediction based on our general framework for text prediction and classification in email. The platform includes components for email processing, binary feature extraction, and a maximum entropy classifier from the MALLET toolkit (McCallum, 2002). General features include the stemmed content and subject of a message as well as document length. We also designed a variety of email specific features such as sender and recipient identity.

5. Experimental Setup

We evaluated our predictor on the spam-free inboxes of two computer science graduate students, Merrick and Jarvis. The email in each dataset represents received messages that went to the user's inbox and not those automatically filtered into other folders upon arrival. Therefore, most automated messages and listserv mail, predominantly negative instances, were automatically excluded.

Merrick and Jarvis each manually sorted through their email and identified messages which needed replies[3]. Merrick's email contained a total of 626 messages of which 210 were *needs reply*. Jarvis's email contained a total of 741 messages, including 281 positive instances. The corpus was a random selection from a time span of approximately one month for Merrick and 5 months for Jarvis.

When testing Merrick-Merrick or Jarvis-Jarvis (within-task) baselines, we used 10 random splits of the corpus into train (80%) and test (20%) sets. When testing Merrick-Jarvis or Jarvis-Merrick (transfer) models, we test on all of Jarvis's or Merrick's data.

6. Feature Design

In this section we discuss both in-domain and transfer results with two types of features. The first type, content-centric features, correspond to standard features on email, like "sender", bag-of-words representation of the subject

[3]One could *infer* the labels by observing which emails were actually replied to in the past. Unfortunately, users often reply to emails that do not need replies and miss emails which do. Furthermore, these inferred labels are quite noisy and difficult to predict.

Content-Centric	*User-Centric*
from Fabrizio[4]	from supervisor
from Tahlia[5]	from frequent contact
to Merrick	Only to me
from Kenrich[6]	address book contact
thread contains word X	active thread

Table 1. Some content-centric features and their corresponding user-centric equivalents. The content-features are mapped to the user-centric features, greatly reducing the dimensionality of the feature space and creating overlap between user feature domains.

User	*Feat*	*Rec*	*Prec*	*F*
Merrick	CC	.47	.61	.53
	UC	.67	.81	**.73**
Jarvis	CC	.54	.61	.57
	UC	.65	.83	**.73**

Table 2. Results for training and testing on hand labels using user-centric (UC) or content-centric (CC) features.

and bodies, and "people in to list". The second type, user-centric, we introduced in section 1 and 2. These features identify patterns of behavior common to multiple users and useful for the task of reply prediction.

Table 1 identifies some common content-centric features and their corresponding user-centric features.

6.1. Empirical Results

We performed two evaluations. In the first, we trained and tested classifiers classifiers on Merrick and Jarvis independently. This test investigated the predictive value of user-centric features. The second test transfered the trained classifiers between users. This evaluation investigated the transfer value of user-centric features. Each test was repeated for both user-centric and content-centric features.[7]

As expected, modeling user behavior yielded a large increase in performance for each user. As shown in Table 2, Merrick's classifier increased 20 points and Jarvis's classifier 16 for user-centric features. These results clearly demonstrate the predictive value of user-centric features. Our transfer learning test showed a dramatic reduction in loss after transfer. When transferring from Jarvis to Merrick, performance dropped 4 points while transfer from Merrick to Jarvis dropped only 2 points. This validates our belief of the transfer potential of user-centric features. Surprisingly, the transferred classifier in both cases performed nearly as well as a classifier trained on the target user. (Table 3)

[7]We also tested classifiers using both user-centric and content-centric features but its performance did not differ significantly from the user-centric classifiers.

Train	*Test*	*Feat*	*Rec*	*Prec*	*F*	Δ
Mer	Jar	CC	.31	.62	.41	-.16
		UC	.60	.88	**.71**	**-.02**
Jar	Mer	CC	.27	.49	.35	-.18
		UC	.70	.68	**.69**	**-.04**

Table 3. Transfer learning using hand labeled features. Each classifier is trained on one user and then tested on the other. Two classifiers were trained for each user, a content-centric (CC) and a user-centric (UC). The delta shows the difference between the transfered classifier and the best classifier (trained and tested on the same user). The delta is obtained from comparison with Table 2.

User	*User-Centric*	*Content-Centric*
Merrick	157	16,414
Jarvis	159	11,171

Table 4. Total features for each user. Despite there being far fewer user-centric features, they still yield a much higher performance.

6.2. Discussion

It is clear from our results that user-centric features represent a powerfully predictive set of features that enable transfer learning. Despite their much smaller number, user-centric features are still able to outperform content features (Table 4). Further evidence of the power of user-centric features comes from their dominance among the most predictive features. We trained a classifier on Merrick's data with both user-centric and content-centric features. Over the ten randomized runs, user-centric features were assigned the highest weights, comprising an average of 9.2 out of the top 10 most predictive features.

Since user-centric features model general behavior, we would expect that two classifiers trained on different users using user-centric features would show a high agreement level compared to a content-centric classifier. We compared the predictions of both Jarvis- and Merrick-trained classifiers on 10 test subsets and measured the agreement between the two predictions. Content-centric classifiers showed an agreement level of 71% on Merrick's data and 70% on Jarvis's. In contrast, the user-centric classifiers agreed 86% of the time on Merrick's data and 92% on Jarvis's. The high agreement demonstrates that by modeling the behavior, each classifier can learn the general behavior of the domain, which is more transferable than task-specific models.

Finally, we return to section 3. There, we showed that by designing a representation $\mathcal{R}$ which makes two tasks distributionally similar while still being an effective representation for classification, one can decrease transfer generalization error. This section demonstrates empirically that user-centric features are exactly such a representation.

7. Conclusions and Future Work

We presented an analysis of feature design for transfer learning. We demonstrated theoretically that designing features which both make two tasks appear similar and are good features for classification can decrease "transfer" generalization error in a new task. Email reply prediction is a task where standard content-centric features differ significantly from user to user. We showed how to design user-centric features which model user behavior, rather than specific properties like sender, recipients or content bag-of-words features. These user-centric features empirically verify our theory, providing a large gain for transfer when compared with content-centric features.

Our current work focuses on feature design, but this can be difficult and time-consuming. Furthermore, manual design invariably misses many features which may be useful for transfer. An important future goal is to learn deictic features automatically from a set of content-centric features. Rather than hand crafting these features for every domain, a learning system would induce meta-features, features that described and model the content-centric features in a more general way.

One technique for doing this is to map one or more content features to a shared representation, which in turn functions as a vehicle for task transfer. A well-known method for inducing such representations is a shared hidden structure similar to the one outlined by Caruana (1997). Similar representations were explored in more empirical and theoretical detail by (Ando & Zhang, 2004; Krupka & Tishby, 2005). We are currently developing methods for learning deictic features that build on these ideas.

References

Ando, R. K., & Zhang, T. (2004). A framework for learning predictive structures from multiple tasks and unlabeled data. *Technical Report RC23462, IBM TJ Watson Research Center*.

Anthony, M., & Bartlet, P. (1999). *Neural network learning: Theoretical foundations*. Cambridge University Press.

Baxter, J. (1995). Learning internal representations. *COLT '95: Proceedings of the eighth annual conference on Computational learning theory* (pp. 311–320). New York, NY, USA.

Ben-David, S. (2003). Exploiting task relatedness for multiple task learning. *COLT 2003: Proceedings of the sixteenth annual conference on Computational learning theory*.

Boone, G. (1998). Concept features in Re:Agent, an intelligent email agent. *Proc. of the the 2nd International Conference on Autonomous Agents* (pp. 141–148). St.Paul, MN.

Caruana, R. (1997). Multitask learning. *Machine Learning, 28*, 41–75.

Cohen, W. W., Carvalho, V. R., & Mitchell, T. M. (2004). Learning to classify email into "speech acts". *EMNLP*. Barcelona, Spain.

Crammer, K., Kearns, M., & Wortman, J. (2005). Learning from data of variable quality. *Neural Information Processing Systems (NIPS)*. Vancouver, Canada.

Dabbish, L., Kraut, R., Fussell, S., & Kiesler, S. (2005). Understanding email usage: Predicting action on a message. *CHI '05*. Portland, OR: ACM Press.

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the Query By Committee algorithm. *Machine Learning, 28*, 133–168.

Krupka, E., & Tishby, N. (2005). Generalization in clustering with unobserved features. *NIPS*. Vancouver, Canada.

McCallum, A. (2002). MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Neustaedter, C., Brush, A. B., Smith, M. A., & Fisher, D. (2005). The social network and relationship finder: Social sorting for email triage. *Conference on Email and Anti-Spam (CEAS)*. Mountain View, CA.

Sutton, C., & McCallum, A. (2005). Composition of conditional random fields for transfer learning. *HLT/EMLNLP*.

Whittaker, S., & Sidner, C. (1996). Email overload: exploring personal information management of email. *CHI '96* (pp. 276–283). Vancouver, British Columbia, Canada: ACM Press.