An Adaptive Smoothing Method for Sensor Noise in Augmented Reality Applications on Smartphones

Rifat Ozcan[1,*], Fatih Orhan[1], M. Fatih Demirci[2], and Osman Abul[2]

[1]AnelARGE, Hacettepe Teknokent, Ankara, Turkey
{rifat.ozcan,fatih.orhan}@anelarge.com
[2]Computer Eng. Dept., TOBB University of Economics and Technology, Ankara, Turkey
{mfdemirci,osmanabul}@etu.edu.tr

Abstract. Handling inaccurate and noisy sensor readings are among important challenges while implementing augmented reality applications on smartphones. As a result, we need to smooth the sensor readings for steady operation. However, no smoothing algorithm performs best in all cases as there is an inherent tradeoff. On one hand, excessive smoothing slows down the effect of device movements, hence makes applications less responsive. On the other hand, insufficient smoothing causes objects on the screen to constantly move back and forth even while the device is steady, hence makes applications too responsive. Clearly, both of the extremes cause augmented reality applications to be less effective in terms of human-computer interaction performance. In this paper, we propose an adaptive smoothing method based on the rate of change in device view direction. Basically, the method adjusts the smoothing level adaptively based on the phone movement. Our experimental results show that our adaptive approach, in comparison to previous proposals, achieves a better smoothing for various cases of phone movements.

Keywords: Augmented reality, sensors, noise, smoothing.

1 Introduction

An Augmented Reality (AR) system combines real and virtual objects in a real environment, registers (aligns) real and virtual objects with each other, and runs interactively, in three dimensions, and in real time [1, 2]. Even though initial work on AR systems started in 70's, functional and practical systems appeared in 90's [7]. These elements required for AR systems, such as displays, sensors, batteries and computers were too bulky, heavy and expensive for mobile or everyday use. Starting with the new millennium, the mobile device display units enhanced and grew in size and capability. Moreover, variety of sensors, cameras, GPS receivers, accelerometers, and magnetometers were integrated into smartphones. The sensor integrations and other enhancements provided the necessary components for the realization of AR systems on smartphones [11, 3]. After the introduction of iPhone and Android

* Corresponding author.

N. Venkatasubramanian et al. (Eds.): Mobilware 2011, LNICST 93, pp. 209–218, 2012.

devices, the concept became hype and many applications popped-up one after another. Layar[1] and Wikitude[2] are two popular AR applications among many others. In our company, we also developed a similar AR browser that mainly presents points of interests (POIs) (i.e., restaurants, clubs, cinemas, pharmacies etc.) around user location, and capable of automatic POI filtering based on user context.

An important problem in AR is to track the user or object movements so that virtual and real objects can be aligned properly. This is also referred as the registration problem. The methods can be categorized into two: (i) computer vision based approaches [6], and (ii) sensor based approaches. Computer vision based approaches track the manually placed markers in the scene and determine the position and orientation of the camera. Clearly, this prevents AR systems to be used in outdoor applications. This approach is not well-suited to smartphones, since limited memory and CPU processing capabilities of these devices may not allow costly computer vision based tracking methods to be applied in real time. For the mentioned reason, AR systems on smartphones use the sensor based approaches. Magnetic field and accelerometer sensors together with GPS receivers are used to determine the orientation of the camera. Even though this approach does not require manual markers and enables outdoor usage, noise in mobile phone's magnetic field and accelerometer sensors adversely affect the quality of AR applications, and this fact adds another challenge into the problem [5].

In the literature, various smoothing techniques are applied in different fields, including signal processing [8, 10], statistics [9], and information retrieval [12, 4] in order to remove noise from the data. These methods vary in complexity and accuracy. To the best of our knowledge, Gotow et al. [5] presents a pioneering work towards this problem in the context of AR systems and smartphones. The authors propose *compass filter algorithm* to smooth the readings from magnetic field sensor. This algorithm identifies outliers in the noisy sensor readings based on a deviation threshold parameter. If the deviation of a new sensor measurement from the mean of sample is higher than the deviation threshold, then this measurement is interpreted as outlier.

In this paper, we propose an adaptive smoothing method for sensor noise cancellation. The main contributions of this study are as follows:

- We first analyze the noise in sensor readings for various cases of phone movements such as phone-stationary at hand and phone-rotating about an axis.
- We propose an adaptive smoothing method which adjusts the level of smoothing to the phone movement. This is achieved by monitoring the change in phone view direction.
- We show in our experiments that the proposed approach provides us with far better results over the three previous methods, simple, exponential, and compass filter [5] smoothing algorithms.

The rest of this paper is organized as follows: In the next section, we present the alternative smoothing methods in the literature, which are applicable to smartphones.

[1] http://www.layar.com/
[2] http://www.wikitude.org/

This section also introduces our adaptive smoothing method. Later, we evaluate and compare the proposed approach and alternative smoothing algorithms in the experimental section. We close the paper with our conclusions.

2 Smoothing Methods

In this section we present three smoothing approaches (namely, simple moving average, exponential moving average and compass filtering algorithm) that are applicable to smartphones from the literature, and describe our adaptive smoothing method afterwards.

In a typical mobile augmented reality application, the objective is to present the nearby points of interests (POIs). This requires a mapping of 3D object locations to the 2D smartphone screen. It is trivial to find nearby POIs given the locations of the device and POIs. However, the exact location of POIs on the smartphone screen changes based on the phone orientation. The orientation is computed using the values read from magnetic field and accelerometer sensors. However, the noise in these sensors causes the location of the POI to be noisy which in turn results in POI to be going back and forth even though the device is steady at hand. This causes a frustrating user interface especially when the user wants to click the POI for further detailed information.

We apply smoothing for magnetic field and accelerometer sensor values using a history of measured values. Let S_i to be the i^{th} measurement from a sensor (For the sake of simplicity, we will assume one sensor measuring one value. It is straightforward to extend this to the case where a sensor giving a vector in 3D.). We define the history array of sensor measurements as follows, where HL denotes the length of the measurement history array:

$$H = [S_0, S_1, \ldots, S_{HL-1}]$$

Each time we get a new value from the sensor, the history array shifts by one position; causing the least recent measurement (in this case, S_0) to be forgotten, and the new measurement to become the most recent measurement (in this case, S_{HL-1}). Finally, the smoothed value is computed based on the smoothing function (f_s) and history array (H).

2.1 Simple Moving Average

This approach simply takes the mean of the recent measurements as follows:

$$f_s = \frac{\sum_{i=0}^{HL-1} H[i]}{HL} \tag{1}$$

The drawback of simple moving average is that it incurs a significant lag to the latest data point. The lag duration is directly proportional to the length, HL, of the history array. The computation complexity of this approach is $O(1)$ assuming that the current sum is kept and constantly updated.

2.2 Exponential Moving Average

The simple moving average assigns the same weight to all recent data points independent of the time of the measurement. However, in some cases, it is desirable that more recent measurements should have higher weights in comparison to older measurements. Exponential moving average achieves this criterion by giving exponentially decreasing weights to older data points [9]. The formula for this smoothing function is given below:

$$f_s = \alpha \sum_{i=0}^{HL-1} (1-\alpha)^{((HL-1)-i)} H[i] \tag{2}$$

The drawback of this method is the need to tune the parameter alpha, a fraction between 0 and 1. The value of alpha needs to be close to 1 when phone is rotating and making sharp moves, but when the phone is stationary choosing a value closer to 0 is better as this selection assigns the same weight to all history values. Therefore, no magic alpha value can cover all the use cases. Moreover, the method is relatively costly, especially on smartphones, since the formula requires many floating point operations.

2.3 Compass Filtering Algorithm

Gotow et al. [5] discuss that noise reduction in smartphone sensors as one of the challenges for augmented reality applications, and as a result they propose a custom smoothing algorithm. The main idea behind their method, called compass filtering algorithm, is to identify outliers in the noisy sensor values. The outliers are detected based on a deviation threshold parameter. If the deviation of a new sensor measurement from the mean of sample is below the deviation threshold, then this data is interpreted as a non outlier, and inserted into the data buffer. Otherwise, it is tagged as outlier and put into another buffer, called outlier buffer. The simple average of values in the data buffer is returned as the smoothed sensor value (see [5] for the details of the algorithm).

2.4 Adaptive Smoothing Method

Our smoothing method is adaptive in the sense that the rate (level) of smoothing is dynamically adapted to the smartphone rotation. The basic idea is to use prolonged history values when the device is stationary and not rotating, and otherwise focus on only the most recent values. This achieves the smoothing of noisy values and prevents POIs to be appearing going back and forth constantly even the device is stationary at hand. If the phone is rotating or making sharp moves, we use fewer history values in order to respond to the rapid changes in the POI location in a reasonable time so as to diminish the lag time as much as possible. We first describe how we identify that the phone is rotating, and then present the details of our algorithm.

For our method, we need to introduce rotation matrix (R) which is calculated based on values from magnetic field and accelerometer sensors (Note that this rotation

matrix performs the projection of object location in world coordinates to phone coordinate system). We re-compute R each time we have a new sensor reading. The third row of a rotation matrix gives the phone view direction of the camera. We call this row vector as the view vector (V). Suppose that R_p denotes the rotation matrix computed in the previous sensor update, and V_p is the view direction based on R_p. Then, the rate of change (C) in the view direction is computed by the magnitude of the view difference vector computed as follows:

$$C = \left| \vec{V} - \overrightarrow{V_p} \right|$$

We measure the rate of the change value for different phone movements such as phone-stationary at hand, and phone-rotating about an axis. More details on this will be provided in the next experimental section. In the light of our experiments, we determine a value, C_{thr}, for the threshold parameter. If the rate of change value is below this threshold then the phone is predicted to be stationary, otherwise it is assumed rotating. Our smoothing algorithm is based on this decision.

The main idea in our method is to keep a virtual history length (VHL) that changes based on the phone movement. Whenever we read raw sensor values M_{raw} and A_{raw}, from magnetic field and accelerometer sensors, respectively, we store these values in history arrays H_{Mag} and H_{Acc}. We determine the phone rotation based on threshold value (C_{thr}) and rate of change (C) computed as described previously. If the device is predicted as stationary, we increase VHL by α, otherwise we decrease it by β. Finally, we restrict the value of VHL to change between a lower, HL_{min}, and an upper, HL_{max}, bound. Then, the sensor value is smoothed by the simple moving average of last VHL values in the respective history array (H_{Mag} or H_{Acc}). This way, we achieve an adaptive smoothing based on phone movement.

3 Experiments

To demonstrate our approach to noise removal in an AR application, we experiment with an HTC G1 smartphone running Android operating system. The phone has a magnetic field sensor and an accelerometer sensor. In the experiments, we configure sensors in order to get the data as fast as possible. According to our measurements, in this mode, magnetic field and accelerometer sensors output about 50 and 40 readings per second, respectively. In the following, we first present the experimental results for the detection of the phone rotation. Then, we compare three smoothing algorithms against our method.

Our experiments consist of four different phone movement scenarios. First, we leave the phone steady on the table and read the sensor values. Next, we hold the phone at hand but do not move it at all. In the third and fourth cases, we rotate the phone about X and Y axis, respectively around ±90 degrees. The experimental results obtained with this setup are presented in the following. We present figures for only two most representative and relevant of these four cases, namely phone is steady at hand and phone is rotating about Y axis (Other two cases show similar trends).

3.1 Detection of the Phone Rotation

In this section, the objective is to measure the rate of change value (*C*) in the view direction for different phone movements. We empirically fixed the value for the threshold parameter C_{thr}. The plot in Fig.1a shows the value of *C* when the smartphone is steady at hand. In this case, the values are noisy and the value of *C* varies considerably. However, we observe that there is a minor movement while the phone is at hand, therefore the absolute value may exceed 0.0020. Fig.1b shows the case when the phone is rotated about Y axis. The results show that *C* increases up to the range [0.015-0.020]. The reason for the decrease is that we first rotate the phone in some direction and we stop at some point for a while, and then rotate the phone back to its original orientation, i.e. like a swing move.

The experimental results are used to estimate the threshold value for the rate of change in the view direction. In particular, this value is set to 0.0020 such that if *C* is less than this value, the device is predicted to be stationary. Otherwise it is rotating.

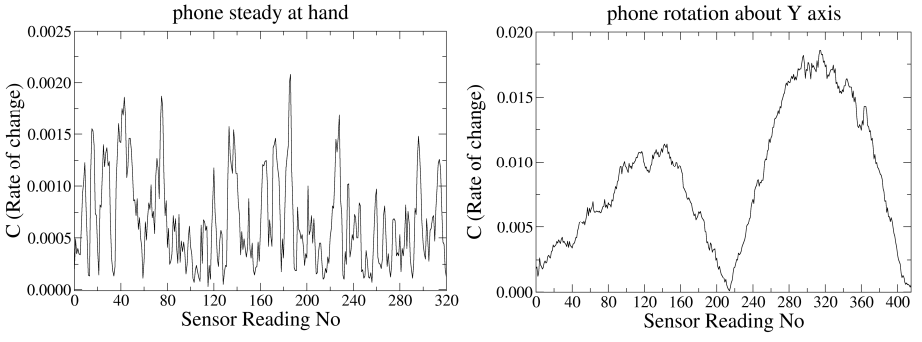


Fig. 1. Rate of change in phone view direction when a) phone is steady at hand, and b) phone is rotating about Y axis

3.2 Comparison of Smoothing Methods

In this section, we first evaluate our method for different phone movement scenarios. Then, we compare our approach against the other smoothing techniques. In the experiments, we set the value of HL_{max} to 120 (representing roughly two seconds of sensor data) in order to ensure that when the phone is stationary, and thus the POIs will not move back and forth. We do not allow the *VHL* value to decrease to one so that it does some sort of smoothing but only relies on last 20 values (HL_{min}). It is observed that considering the last 20 sensor values does not cause any distracting delay for users. The threshold value for view vector change (C_{thr}) is experimentally tuned and found to be 0.003 which gives the best case. We experimented with different values for α and β parameters. If we set α (the increase factor for *VHL* if we detect that the phone is stationary) to a value bigger than 1, it is observed that the sensor value fluctuates even though the device is kept stationary. We conjecture that

the reason for this behavior is that when we increase the *VHL* value larger than 1, this causes the history values to include raw sensor values that belong to the time when the phone is rotating. This causes the smoothed value to be fluctuating even though the device is stationary for some time. Therefore we set both α and β parameters to 1 in order to handle this case. This ensures that when we increase the *VHL* value, we always grow the history array towards the new values (future direction) instead of old values (past direction).

Fig2 presents how the proposed approach performs for smoothing original sensor values. In particular, Fig. 2a shows the smoothing when the phone is stationary at hand. The curves named "original" and "adaptive" correspond to the original and our smoothed sensor values, respectively. We also show the virtual history length (VHL) values based on the secondary Y axis on the right side of the plot. As it is expected, VHL values remain to be the maximum history length (HL_{max} = 120), and our smoothing algorithm stabilizes the sensor values as much as possible. This achieves POIs to be left almost stationary on the phone screen without affected by the sensor noise. Fig. 2b shows the case when the phone is rotating about Y axis. We see that VHL value peacefully responds to the change in sensor values and adapts the smoothing accordingly. Note that there are peaks in VHL values in Fig. 2b. This is due to the fact that, as we mention previously, when we rotate the phone about an axis for some degree, we stop at that point for a while, and then rotate back to the original position. This causes the change in sensor values (so does the change in phone view direction) to remain minimal for that small time period. This increases the VHL value for a while (since the phone is stationary at that time) but later the change becomes larger than the threshold and VHL recovers itself.

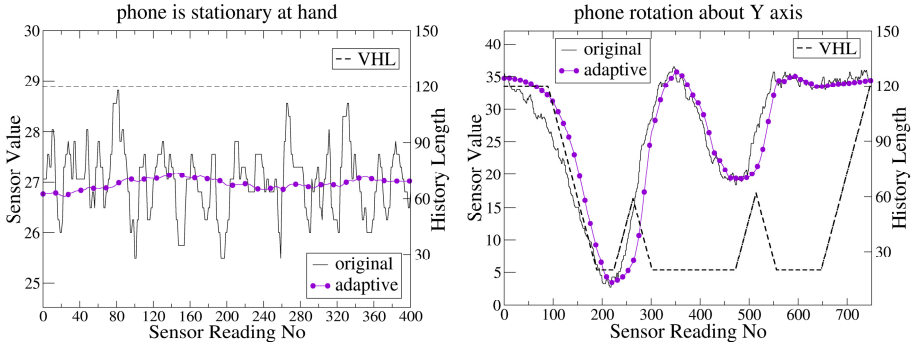


Fig. 2. Our smoothing algorithm and virtual history length (VHL) value when a) phone is steady at hand; and b) phone is rotating about Y axis

Secondly, we compare our smoothing algorithm with simple moving average and compass filter algorithms. The results are plotted in Fig. 3. Note that compass filter algorithm proposed in [5] requires two parameters: a) data buffer length (like history length in some sense), b) the deviation threshold to determine outliers. We examine

with different parameter values, and observe that it behaves like a simple moving average method in the sense that if you increase the value of data buffer size, it makes more smoothing but when the phone is rotating it incurs more delay as depicted in Fig. 3. We set the buffer size to 60 and the deviation threshold value to 1 for compass filter smoothing. In order to show the delay due to smoothing, we draw drop lines in the plot where the sensor value reaches to its maximum after the rotation (Points numbered as 1, 2, 3, and 4 show the cases for original sensor value, our adaptive smoothing, compass filter and simple moving average, respectively). We observe that our adaptive approach incurs the shortest delay due to smoothing while simple moving average has the longest delay. The smoothing delay can be computed quantitatively as follows: For example, the number of sensor readings between points 1 and 2 is divided by the number of sensors readings per second (in our case, this is approximately 90 values per second, for both sensors, in total). This computes the delay due to smoothing. According to our measurements, the delay between original and our adaptive method is only 130 msec, while compass filter and simple moving average smoothing incurs 330 msec and 800 msec delays, respectively. Note that this same behavior is also observed when sensor value drops to its local minimum points (sensor readings around 200-250, and around 500-550). In this way, our AR application can respond to phone rotations quickly and change the POI locations without any distracting delay. Furthermore, we repeat this experiment 10 times and observe the same typical trend as in Fig.3.

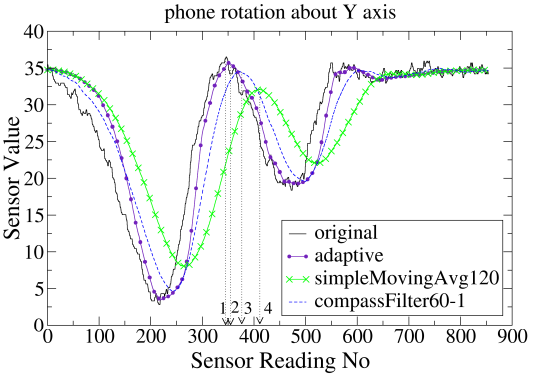
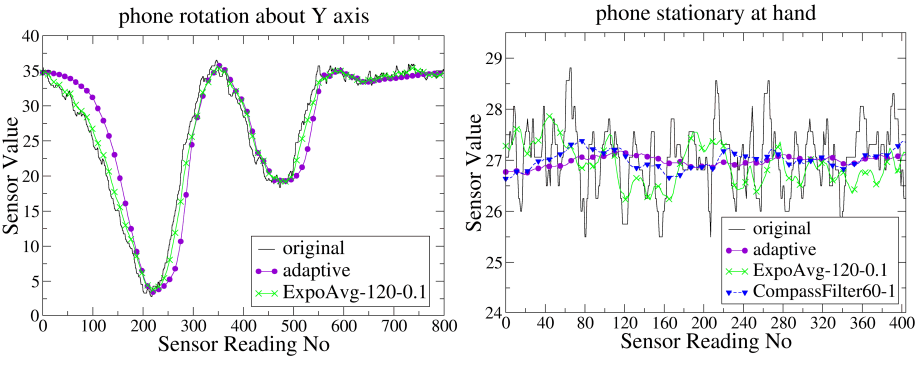


Fig. 3. Comparison of smoothing methods when the phone is rotating about Y axis. We draw drop lines in the plot in order to show the delay due to smoothing. Points numbered as 1, 2, 3, and 4 show the cases for original sensor value, our adaptive smoothing, compass filter and simple moving average, respectively. Our smoothing approach incurs the shortest delay while simple moving average has the longest delay.

Finally, we compare our approach with exponential smoothing method in Fig. 4. The plot in Fig. 4a, shows the case when the phone is rotating about Y axis. As we previously mention, the alpha value in exponential smoothing must be tuned for different cases. For example, we tune this parameter to 0.1 when phone is rotating and the result seems that exponential smoothing does better than our approach because it

does not incur any delay. However, this value of alpha does not work when phone is stationary as shown in Fig. 4b. In this case, the exponential smoothing algorithm cannot handle the noise in sensor values unlike our algorithm. Therefore, it is impractical to select a single value for alpha which works in all cases. Compass filter smoothing incurs slightly more fluctuation of sensor values than our approach. Note that, when the phone is stationary, simple moving average exhibits exactly the same behavior as our approach. In order to show the smoothing quantitatively when the phone is stationary, we use the linear least squares fit of a straight line to the sensor values (since we expect the sensor values to be almost the same when the phone is stationary as shown in Fig.2.a) and measure the root mean square error from the fitted line. According to the results, the original, exponential smoothing, compass filter smoothing, and the proposed adaptive smoothing have 0.66, 0.36, 0.17, and 0.09 error rates, respectively. The results reveal that our approach achieves far better results compared to the alternative methods.



(a) (b)

Fig. 4. Comparison of our smoothing approach and exponential smoothing when a) phone is rotating about Y axis, and b) phone is steady at hand. It is observed that it is possible to tune the exponential smoothing for the case of phone rotation as in a), but this tuning does not perform well if the phone is stationary as in b). However, our smoothing approach is a good compromise between these two extreme cases and adapts itself to the device movement.

4 Conclusions

Noise reduction in smartphone sensors is one of the challenges for AR applications. In this paper, we propose an adaptive smoothing approach, which is movement-aware in the sense that it adapts the level of smoothing based on the phone movement behavior. We conducted experiments based on real sensor values read from a smartphone for various phone movement cases (e.g. steady at-hand and rotation). Our results show that even though it is possible to tune alternative smoothing techniques to perform well in specific cases, our adaptive smoothing method achieves a better smoothing performance in the overall case. Although our approach was applied on a

HTC G1 phone, we expect the same smoothing performance on other smartphones. In the future, we will evaluate the performance on other smartphones especially for Apple's iPhone. In addition, we plan to provide a general method to determine the value of the C_{thr} threshold as it was determined experimentally in this paper.

Acknowledgments. This work is supported by the Scientific and Technological Research Council of Turkey (TÜBÝTAK) by the grant number 7100183.

References

1. Azuma, R.: A Survey of Augmented Reality. Presence 6(4), 355–385 (1997)
2. Azuma, R.T., Baillot, Y., Behringer, R., Feiner, S.K., Julier, S., MacIntyre, B.: Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications 21(6), 34–47 (2001)
3. Bimber, O., Raskar, R.: Modern Approaches to Augmented Reality. ACM SIGGRAPH 2005 Courses, Article 1 (2005)
4. Chen, S.F., Goodman, J.: An Empirical Study of Smoothing Techniques for Language Modeling. In: 34th Annual Meeting on Association for Computational Linguistics, pp. 310–318 (1996)
5. Gotow, J.B., Zienkiewicz, K., White, J., Schmidt, D.C.: Addressing Challenges with Augmented Reality Applications on Smartphones. In: Cai, Y., Magedanz, T., Li, M., Xia, J., Giannelli, C. (eds.) Mobilware 2010. LNICST, vol. 48, pp. 129–143. Springer, Heidelberg (2010)
6. Hoff, W.A., Nguyen, K., Lyon, T.: Computer Vision-based Registration Techniques for Augmented Reality. In: Intelligent Robots and Computer Vision XV. SPIE, vol. 2904, pp. 538–548 (1996)
7. Krevelen, D.W.F., van Poelman, R.: A Survey of Augmented Reality Technologies, Applications and Limitations. The International Journal of Virtual Reality 9(2), 1–20 (2010)
8. Ngo, T.B., Le, H.L., Nguyen, T.H.: Survey of Kalman Filters and their Application in Signal Processing. In: ICAI 2009, vol. 3, pp. 335–339 (2009)
9. NIST/SEMATECH e-Handbook of Statistical Methods, `http://www.itl.nist.gov/div898/handbook/`
10. Orfanidis, S.J.: Introduction to Signal Processing. Prentice Hall (1995)
11. Schmalstieg, D., Wagner, D.: Experiences with Handheld Augmented Reality. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, pp. 1–13 (2007)
12. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In: SIGIR 2001, pp. 334–342 (2001)