An approach to automatic generation of fuzzy membership functions using popularity metrics

Aitor Almeida[1], Pablo Orduña[1], Eduardo Castillejo[1], Diego López-de-Ipiña[1], Marcos Sacristán[2]

[1] Deusto Institute of Technology - DeustoTech
University of Deusto. Bilbao, Spain
{aitor.almeida, pablo.orduna, eduardo.castillejo, dipina}@deusto.es
[2] Treelogic. Llanera, Spain
marcos.sacristan@treelogic.com

Abstract. Creating membership functions for fuzzy system can be a difficult task for non-expert developers. This is even more difficult when the information available about the specific domain is limited. In our case, we wanted to create membership functions that model the different characteristics of mobile devices. Due to the lack of public data about the mobile phones sales it is difficult to estimate the market share of each device. To tackle this problem we have developed a mechanism that uses popularity metrics to estimate the market share and generate the membership functions. In this paper we describe and discuss the used algorithm.

Keywords: fuzzy; mobile devices; characterization; membership functions; google trends; popularity; WURFL.

1 Introduction

While developing the Imhotep framework [1] for the creation of adaptative user interfaces, one of our design goals was to simplify the development cycle as much as possible. As part of the framework, developers can use preprocessor directives to guide the interface adaptation. These directives employ the device characteristics (screen size, CPU, RAM memory, supported formats, codecs…) to decide how the interface adaptation should be done. One problem we identified during the development of the framework was that developers without extensive experience usually do not have the required knowledge to identify the exact values to be used in the preprocessor directives. For example a developer might want to know if a processor is “fast” or a screen is “big”, without having to deal with specific values. Working with values closer to natural language eases the use of preprocessor directives. We encountered initially several problems with this approach. Obviously one screen is big when we compare it with the other screens in the market. To have an accurate concept of what is “big” we have to know the market share of the devices to identify the distribution of the screen sizes. Also, the concept “big” will change over time. What was considered a big screen in 2004 nowadays would be considered

normal or small. Finally, it will also change with the location; a big screen in Europe probably is not so big in Japan. To be able to create fuzzy membership functions that model the device characteristics accurately we would have needed the market share data of all the existing devices. Being this option completely unfeasible we adopted another approach: the use of popularity metrics to infer the device adoption rate. To do this we have used two different tools. First we have used WURFL, an XML mobile device database, to compile an exhaustive list of the existing devices and their characteristics. With that list we have retrieved the popularity metrics of those devices from Google Trends.

Using this data we have developed a process to automatically generate membership functions. We will describe this process in the following paper. In Section 2 we will analyze the related work, in Section 3 we will explain how the generation process works. Finally in Section 4 we will expose the conclusions and future work.

2 Related Work

Google Trends has been used to model different domains and scenarios. In [2] authors use data from Google Trends and Google Insights to make short-term economic predictions. This approach has also been used to predict private consumption [3]. Another domain where Google Trends has been used is epidemiological research, studying influenza epidemics [4] or the expansion of Lyme disease [5]. In [6] authors present a web tool for disease outbreak surveillance based on Google Trends. Finally in [7] the trend data is used to track diseases. As can be seen the information contained in Google Trends is an interesting pool of data that can be used to model and infer the behavior of the users.

Automatic membership function generation has already been addressed by several authors. This problem has been tackled using different approaches. In [8] and [9] authors describe a method to generate membership functions using genetic algorithms. Authors in [10] propose the use of inductive reasoning for the construction of membership functions. Finally in [11] authors use an ad-hoc method to generate the membership functions.

3 The Characterization Process

There are situations where the crisp values of device characteristics are not suitable to be used directly. For example, the developer may want to show certain video only if the screen of the device is “big” or to use a certain reasoning engine only if the processor capabilities are “high”. The main problem with this scenario is that the concept “big” is not directly related to one value and is a relative value (which implies that what is a big screen today probably won’t be big in 2 years). The goal of our system is to identify new capabilities using the already existing ones and to fuzzyfy them. To do this we have defined a set of fuzzy rules that take as input numeric values from the existing capabilities and create symbolic values for the new ones. An example for the reasoning that takes place in this stage would be: “If the resolution is

big and the screen size is big the video suitability is very high". This reasoning will be modelled with fuzzy rules. The main problem we have encountered using fuzzy rules is that we need to fuzzify the crisp variables encountered in the databases (in our case WURFL). This raises some challenging questions. What do we consider a "big" screen size? How can we identify what characteristics are inherent of the average mobile device? These concepts are relative to the values of other device models. One screen is big if its height and width are larger than the average values of the other models. To answer these questions we would have to know the actual distribution of the market. Our proposed solution is to use popularity metrics to estimate the market share of the devices (in our case, we use Google Trends). Besides, all the device models can not have the same weight in the calculation, not all the device models have sold the same number of units. This is why the most popular models should have more weight during this calculation. In order to calculate the popularity of one device we have to adjust it with its "age". Popularity fades with the passing of time. Users tend to change their mobile phones frequently, drastically altering the perception of what is a big screen from one year to another. While this number does not represent the sale volume, it is often used as an indicator of the interest shown by the consumers in a specific model [12]. Due to the lack of data regarding the real sale volume for most mobile devices, it is one of the few available indicators. This trend value can change drastically from one location to another; the most popular devices are not the same in Japan and Europe. To tackle this problem we support the geolocation of the results to filter them according to the needs of the developers.

The device characterization process can be divided en three different steps: the initial data retrieval, the decay process and the automatic membership function generation. The first step consists on retrieving and formatting all the necessary data. This means to parse the WURFL database and to retrieve the Google Trends data for all the devices. This is a tedious process (it takes days to gather all the data) due to the IP address and account restrictions of Google Trends and must be done in a distributed way. The final result is a database with all the records of the trends for each device taking into account the geolocation.

```
function createBaseUniverse
Inputs:
  TRENDS is an association of key-value pairs where the keys are each distinct value of the target
  device feature, and the associated value is the summation of the trends
  for the devices with that feature value. The keys are ordered incrementally.
  NUM_OF_TERMS is the number of desired linguistic terms
Outputs:
    REGION_LIMITS is the location of the regions in the base universe

TOTAL_TRENDS ← ∑_{i=0}^{n} VALUES(TREND_i)

NUM_OF_REGIONS ← NUM_OF_TERMS - 1
IDEAL_TREND ← TOTAL_TRENDS / NUM_OF_REGIONS
ACCUMULATED_TREND ← 0
REGION_LIMITS ← empty set
CURRENT_VALUE ← first feature value in TRENDS

for CURRENT_REGION = 1 to NUM_OF_REGIONS do
    while ACCUMULATED_TREND <= CURRENT_REGION * IDEAL_TREND do
      CURRENT_TREND ← associated trend in TRENDS to CURRENT_VALUE
      ACCUMULATED_TREND += CURRENT_TREND
      CURRENT_VALUE ← next feature value in TRENDS
    end while
    add CURRENT_VALUE to REGION_LIMITS
end for
return REGION_LIMITS
```

Algorithm 2. Calculation of the base universe

Once we have all the data the next step is to process the trend values to take into account their "age". Older trend values have less weight in the accumulated trend

value of each device, reflecting the transitory nature of the mobile device market. We have implemented two different strategies for the decay: LogarithmicDecay and ModelDecay. The first one uses a logarithmic function to calculate the decay. The value taken for logarithm base is the point where the trends will no longer have any weight in the calculation (5 years in our case) and while the function returns negative values no decay will be applied. Using this decay strategy, five year old trends will not be taken into account and the newer trends will not have any penalty.

The second strategy takes into account the phone plans of the principal telecommunication companies to try to model the mobile phone change cycle among users. We acknowledge that every user does not change its mobile phone at the end of the acquired mobile plan. We use the following values to calculate the decay: for the latest 15 months we take into account the 100% of the trend value.From 16 to 24 months we take into account the 90% of the trend value. From 25 to 36 months we take into account the 40% of the trend value. From 37 to 60 months we take into account the 10% of the trend value. More than 60 months we take into account the 5% of the trend value. We would like to use a more robust model to calculate the decay, but we have found that this is a good approximation. Once we have the processed data and the desired linguistic terms we can automatically generate the membership functions for those terms. The first step is to divide the data in regions (see Algorithm 2) that will mark the point where each membership function will have its highest value. While creating the regions the algorithm seeks to equally distribute the total trend value contained in each membership function, but usually this goal is not achieved in the first iteration.

```
function findBestUniverse
Inputs:
  REGION_LIMITS is the location of the regions in the base universe
Outputs:
  BEST_UNIVERSE is the universe with the best fitness

ALL_UNIVERSES ← every possible permutation generated by moving one step the limits contained in
REGION_LIMITS

for each PERMUTATION in ALL_UNIVERSES do
    if first region of PERMUTATION does not start in 0 then
        remove PERMUTATION from ALL_UNIVERSES
    for each REGION in PERMUTATION do
        if left limit of REGION > right limit of REGION then
            remove PERMUTATION from ALL_UNIVERSES
        elif left limit of REGION < right limit of previous REGION then
            remove PERMUTATION from ALL_UNIVERSES
    end for each
end for each

UNIVERSE_FITNESSES ← empty set

for each PERMUTATION in ALL_UNIVERSES
    PERMUTATION_FITNESS ← ∑_{region∈PERMUTATION} |IDEAL_TREND − ∑_{trend∈region} trend|
    add PERMUTATION_FITNESS, PERMUTATION to UNIVERSE_FITNESSES
end for each
BEST_UNIVERSE ← first PERMUTATION in UNIVERSE_FITNESSES with lowest
PERMUTATION_FITNESS

return BEST_UNIVERSE
```

Algorithm 3. Calculation of the best possible universe

To solve this problem we generate every possible permutation by moving each of the initial region boundary one step to each side. For each universe we calculate its deviation from the ideal one (see Algorithm 3). First we discard inconsistent universes following these rules: A) If the first region in the universe does not start in the 0 point, then that universe is discarded. B) If the left boundary of a region in the universe starts after the right boundary, then that universe is discarded. C)If the left boundary

of a region starts before the right boundary of a previous region, then that universe is discarded.

The second step is to calculate the deviation of each remaining universe. What we seek is to minimize the deviation from the ideal universe, thus, we select the universe with the lowest deviation. Once we have found the best universe we can finally build the membership functions for each linguistic term (see Algorithm 4). To do this we take into account that: A) The region boundaries mark the inflexion point from the ascending and descending curves of a linguistic term. B)The first linguistic term will only have a descending curve that will start in the left boundary of the first region and will end in the right boundary of the first region. C) The last linguistic term will only have an ascending curve that will start in the first boundary of the last region and will end in the right boundary of the last region. D) The ascending curves are calculated accumulating the trend values in a region. E) The descending curves are symmetrical to the ascending curves: dc(x) = 1 – ac(x), where ac is the ascending curve and dc the descending curve.

```
function calculateMembershipFunctions
Inputs:
  BEST_UNIVERSE is the universe with the best fitness
Outputs:
  MEMBERSHIP_FUCTIONS the membership functions for the linguistic terms

LAST_CURVE ← empty set
MEMBERSHIP_FUNCTIONS ← empty set
for each REGION in BEST_UNIVERSE do
    ASCENDING_CURVE ← ac(x) = ∑_{i=left_limit}^{x} TREND_i | x ∈ feature values in REGION
    DESCENDING_CURVE ← dc(x) = 1 - ac(x) | x ∈ feature values in REGION
    CURRENT_CURVE ← LAST_CURVE + DESCENDING_CURVE
    add CURRENT_CURVE to MEMBERSHIP_FUNCTIONS
    LAST_CURVE ← ASCENDING_CURVE
end for

add LAST_CURVE to MEMBERSHIP_FUNCTIONS

return MEMBERSHIP_FUNCTIONS
```

Algorithm 4. Calculation of the membership functions

4 Conclusions

In this paper we have presented a mechanism to automatically create membership functions using popularity metrics. We have also shown the results of this process, comparing the results of different mobile phones and different locations, and showing how the passing of time changes the relative perception of the characteristics of the devices. As future work we would like to implement some new decay functions and compare the results with the existing ones. We will also implement a new version of the membership function generation mechanism that will allow users to specify the percentage of the trends contained in each linguistic term.

Acknowlegment

This work has been supported by project grant TSI-020301-2008-2 (PIRAmIDE), funded by the Spanish Ministerio de Industria, Turismo y Comercio. We would like to thank the members of our laboratory that helped us recovering the trend data.

References

[1] Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego Lopez-de-Ipiña, Marcos Sacristan. *Imhotep: an approach to user and device conscious mobile applications* Personal and Ubiquitous Computing (Journal), Volume 15, Number 4, pages 419-429. Springer. ISSN: 1617-4909. DOI: 10.1007/s00779-010-0359-8. April 2011.

[2] Varian, Hal R. and Choi, Hyunyoung, Predicting the Present with Google Trends (April 2, 2009). Google Research Blog http://googleresearch.blogspot.com/2009/04/predicting-present-with-google-trends.html. Available at SSRN: http://ssrn.com/abstract=1659302

[3] Schmidt, T. and Vosen, S. Forecasting Private Consumption: Survey Based Indicators Vs. Google Trends. RUB, Dep. of Economics. 2009.

[4] Ginsberg, J., Detecting influenza epidemics using search engine query data. Nature, vol. 457, number 7232, pages 1012—1014, 2008.

[5] Seifter, A. and Schwarzwalder, A. and Geis, K. and Aucott, J. The utility of “Google Trends” for epidemiological research: Lyme disease as an example. Geospatial Health, vol 4, num 2, pp 135-137. 2010.

[6] Carneiro, H.A. and Mylonakis, E. Google trends: a web-based tool for real-time surveillance of disease outbreaks. Clinical infectious diseasesvol 49, num 10, pp 1557. 2009

[7] Valdivia, A. and Monge-Corella, S. Diseases Tracked by Using Google Trends, Spain. Emerging Infectious Disease vol 16, num 1, pp 168. 2010.

[8] Homaifar, A. and McCormick, E. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. Fuzzy Systems, IEEE Transactions on, vol 3, num 2, pp 129-139. 1995.

[9] Shimojima, K. and Fukuda, T. and Hasegawa, Y. Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. Fuzzy sets and systems, vol 71, num 3, pp 295-309. 1995.

[10] Kim, CJ and Russell, BD. Automatic generation of membership function and fuzzy rule using inductive reasoning. Industrial Fuzzy Control and Intelligent Systems, 1993., IFIS'93., Third International Conference on, pp 93-96. 1993

[11] Nieradka, G. and Butkiewicz, B. A method for automatic membership function estimation based on fuzzy measures. Foundations of Fuzzy Logic and Soft Computing, pp 451—460. 2007.

[12] Xu, Kaiquan; et al, "Predict Market Share with Users’ Online Activities Data: An Initial Study on Market Share and Search Index of Mobile Phone" (2010). PACIS 2010 Proceedings. Paper 30.