ESADE
Business School
Ramon Llull University

MASTER THESIS

Topic modeling of microblogging data in tourism

Sebastian Birk
ESADE Business School
M.Sc. in Business Analytics
Academic Year 2017-2018

Tutor:
Albert Armisen Morell

Research Project:
P185 - The role of online emotions and topics in tourism

Barcelona, 8.10.2018

Abstract With the launch of Web 2.0 applications and particularly social media websites in the 2000s, user-generated content has become a crucial factor in the tourism domain both for the decision making of tourists and for effective business management. From a research perspective, this content opens up completely new opportunities to analyze and understand trends and mechanisms in the tourism industry. Aiming to provide a technical foundation to leverage these opportunities, this research study will explore different models based on the topic model method with a particular application focus on the tourism domain. The objective will be to utilize one specific form of user-generated content – microblogging content – to extract a temporal evolution of tourist topics characterized by their geographical presence. To this end, a dataset capturing tweets in Barcelona in the period of time between June and December 2017 will be processed. Latent Semantic Indexing, Latent Dirichlet Allocation, Hierarchical Dirichlet Processes and Non-negative Matrix Factorization will be applied as topic modeling approaches and evaluated quantitatively and qualitatively. It will be investigated whether an automatic detection of the development of microblogging topics over time can be achieved without human supervision, thus providing useful insights into sector trends. The results of these topic modeling approaches can potentially be used to provide decision support for management, recommendations to online tourists and to build a basis for further analytics. The contribution of this research paper therefore consists of the development and application of models based on the topic model method that can extract the temporal evolution of topics of microblogging data. The outcomes of this work will help stakeholders involved in the tourism industry to create value from the enormous amount of social media data that is continuously collected in real time. Furthermore, researchers could make use

of the models to analyze microblogging data in other domains different than the tourism domain.

Keywords Topic Modeling – Mixture Models – Hierarchical Dirichlet Processes – Latent Semantic Indexing – Latent Dirichlet Allocation – Non-negative Matrix Factorization – e-Tourism – Topics in Tourism – Tourism in Barcelona – Twitter & Tourism – Social Media Data – Microblogging – Natural Language Processing – Machine Learning – Data Analytics

Table of Contents

List of Abbreviations

BoW	–	Bag of Words
DM	–	Dirichlet Mixtures
HDP	–	Hierarchical Dirichlet Processes
LDA	–	Latent Dirichlet Allocation
LSI	–	Latent Semantic Indexing
ML	–	Machine Learning
NLP	–	Natural Language Processing
NMF	–	Non-negative Matrix Factorization
SVD	–	Singular Value Decomposition
TFIDF	–	Term Frequency-Inverse Document Frequency
UGC	–	User-Generated Content
WWW	–	World Wide Web

List of Figures

List of Tables

Acknowledgements

This research was part of the INVITE research project (TIN2016-80049-C2-1-R and TIN2016-80049-C2-2-R) funded by the Spanish Ministry of Economy and Competitiveness. I want to thank my tutor Albert Armisen for giving me the chance to participate in this research project and contribute to it by applying and expanding my knowledge about topic modeling and natural language processing. Furthermore, I also want to thank the faculty of “Operations, Innovation and Data Sciences” of ESADE Business School for providing me with the Twitter dataset and the geospatial data of Barcelona used in this research study. Ultimately, I also want to thank my friend and fellow Julian Fischer to support me in the qualitative assessment of the applied topic models.

1 Introduction

1.1 The Potential of Social Media Data Analytics

With the development of the World Wide Web (WWW), considerable bargaining power in many industries has been transferred from suppliers to consumers and suppliers have come under increasing competitive pressure to take active measures to retain existing and acquire new customers (Akehurst, 2009). As a result, suppliers have shown an increasing interest in understanding consumer behavior and have attempted to tailor their offerings to better fit customers' needs. One of the measures that suppliers have taken in this regard during the last two decades is to leverage business analytics and the vast amount of data that has become available through technological advancements accompanying the fast-paced development of the WWW. Already in 2011, the Bloomberg Businessweek found in a survey that 97% of companies with revenues exceeding $100 million were found to use some form of business analytics (Chen, Chiang, & Storey, 2012). Gantz and Reinsel (2012) explored the future potential of business analytics and asserted that by 2020 over 40 Zettabytes (or 40 trillion Gigabytes) of data will have been generated and consumed.

In fact, the amount of data has grown exponentially, thus opening up enormous potential for data analysis. Among the major contributors for this rapid accumulation of data have been social media platforms like Facebook, Twitter and LinkedIn that came with the widespread adoption of Web 2.0 applications in the 2000s. While the era of the Web 1.0 was characterized by a set of static websites where people passively viewed content generated by only a few parties, social media platforms transformed the Internet from a purely informational source to a platform for opinion-sharing (Dippelreiter et al., 2008;

Schmallegger & Carson, 2008; Thevenot, 2007). Consumers can, and increasingly do, share experiences directly with other consumers through electronic word of mouth, and user-generated content (UGC) and peer-to-peer web applications are even perceived to be more credible and trustworthy than traditional marketing communications (Akehurst, 2009). One essential aspect of these platforms is that the opinion of peers can exert a social influence and act as a major lever in the decision-making process of a customer – a topic that has always played a major role in the research field of consumer behavior (Dahl, 2013)[1]. As a consequence, data mining of unstructured text data provided on these platforms has evolved as an important field in the area of data analytics to better understand consumer behavior (Barbier & Liu, 2011). Particularly in industries that are characterized by a widespread use of the Internet as information source and where electronic word-of-mouth is a decisive factor for the selection of goods and services, data analytics should be leveraged to extract insights about consumers and gain competitive advantage (Xiang, Magnini & Fesenmaier, 2015).

Among the social media sites mentioned earlier, above all the microblogging service Twitter has attracted the attention of researchers as it is very well suited for analytical purposes and will thus also be used for this study. Already in 2014, people around the world sent 400 million tweets per day (Walker, 2014) and the service was increasingly used to distribute electronic word-of-mouth, for example about service quality (Sotiriadis & Van Zyl, 2013). Due to the nature of microblogging, the large amount of text in Twitter may presumably contain useful information that can hardly be found in traditional

[1] In this context, social influence means how others affect our awareness, our emotions and our opinions by sharing content about their own experiences and thereby, ultimately, influence our consumption behaviors.

information sources (Zhao et al., 2011). It is, therefore, without a doubt that understanding the informational content of online communication and interactions among citizens and institutions on this platform can, in general, provide significant value for any business or other stakeholder that seeks to understand consumers' behaviors and should thus be a priority of research.

1.2 The Potential of Social Media Data Analytics in the Tourism Domain

Another key insight deduced from previous research is that social influence is particularly important for goods classified as experience goods (Klein, 1998). Experience goods are goods whose quality is usually unknown before consumption (Nelson, 1970). This results in the need for consumers to rely on word-of-mouth and online content to make inferences about such goods. One of the industries in which most services and products offered are experience goods is the tourism industry, which will be the focus of this study. The quality of tour operations and hotels, for example, is only known after the service has been consumed (Litvin, Goldsmith & Pan, 2008). Tourism can, thus, be described as an information-intense industry (Werthner & Klein, 1999); therefore, it is critical to understand changes in information sources that impact the distribution and accessibility of travel-related information and that could affect consumer behavior. Particularly, it has been argued that understanding the nature of the online tourism domain, i.e., the composition of online tourism related information potentially available to travelers, provides an important stepping-stone for the development of successful marketing programs and better information systems in tourism (Fesenmaier, Wöber & Werthner, 2006; Xiang, Wöber & Fesenmaier, 2008). Other research findings also thoroughly demonstrate the

strategic importance of social media content for tourism competitiveness (Leung, Law, van Hoof & Buhalis, 2013).

In addition, specific services related to tourism are often only consumed once by a person and the only way to gather experience before the decision-making process is to rely on the experience of other people. Indeed, as tourism-related products and services are high-priced, imply high involvement, and are well-differentiated in nature, travelers generally collect and review various forms of travel experiences of peers early in the travel decision-making process to minimize the risk of making wrong decisions (Jeng & Fesenmaier, 2002). This fact again emphasizes the importance of online content in the tourism sector. As a logical consequence, a substantial body of academic literature revolving around the topic of tourism in the internet has emerged since the late 1990s (Liu, 2005).

In accordance with the research conducted in this paper, previous research in the domain of tourism has mostly focused on understanding consumer behavior and consumer preferences. In the past, articles that focused on the behaviors, preferences and perspectives of tourists/visitors accounted for 11% of all articles related to tourism across all popular tourism journals, thus constituting the majority of all investigated subfields and highlighting the relevance of this topic (Ballantyne, Packer & Axelsen, 2009).

One of the major findings in this field of research was indeed that individual decisions are strongly influenced by online evidence of experiences that other people made (Pang & Lee, 2008). It has, for example, been shown that traveler reviews have a significant impact on online sales, with a 10 percent increase in traveler review ratings boosting online bookings by more than 5 percent, thus highlighting the importance of online user-

generated reviews to business performance in tourism. Research from Google has shown that 84% of leisure travelers use the Internet as a planning resource (Howison, Finger & Hauschka, 2010). Furthermore, searching for travel-related information is one of the most popular online activities (Qiang, Law, Gu & Chen, 2011); and past surveys indicate that more than 74 percent of travelers use the comments of other consumers as information sources when planning trips for pleasure (Gretzel & Yoo, 2008). In a study with American adults, between 73% and 87% among readers of online reviews of restaurants, hotels and various services such as travel agencies and doctors reported that reviews had a significant influence on their purchase (Pang & Lee, 2008). In 2013, more than one-third of all leisure travelers in the United Kingdom chose their hotels on the basis of social media sites like TripAdvisor and Facebook (Leung et al., 2013).

Moreover, some recent studies have suggested that social media plays an important role not only for consumers in travel information search but also as a tourism marketing tool and for business management. (Chan & Guillet, 2011; Inversini, Cantoni & Buhalis, 2009; Munar, 2011; Xiang & Gretzel, 2010). Stepchenkova and Morrison (2006), for example, suggested that marketing a country could be problematic without a clear understanding of how a tourist destination is perceived by travelers. Pantelidis' (2010) study demonstrated the importance of monitoring and managing electronic communications on social media. Not only can tourism suppliers achieve a better understanding of what consumers want and how they perceive their companies, customers' comments and opinions can also highlight areas of improvement and enable suppliers to protect their brands and images. Dellarocas (2003) also suggested that social media provide tourism companies with unprecedented opportunities to understand and respond to consumer

preferences. Through analyzing the comments on online communities such as TripAdvisor and Virtualtourist, hotels and other travel-related companies are able to better understand what their guests like and dislike about them and their competitors.

One can, therefore, conclude that knowledge about the informational content of social media can be valuable for both business management and for the creation of better-tailored customer experiences. It can demonstrably be utilized to extract business insights, detect current business trends and understand relevant consumer preferences and behaviors related to tourism. Considering the important role of social media in both travelers' decision making and in tourism operations and management, a plethora of research specifically on the application of social media in tourism and hospitality has been catalogued in referred journals (e.g. Chan & Guillet, 2011; Li & Wang, 2011; Noone, McGuire & Rohlfs, 2011; Xiang & Gretzel, 2010).

While there is an extensive body of literature on the relevance of social media platforms for the tourism domain (e.g. Gretzel, 2006; Pan, MacLaurin & Crotts, 2007), there are only a few studies that conduct empirical research trying to leverage the data available on these platforms. Some research on content analysis of user reviews on popular platforms such as yelp.com or tripadvisor.com can be found (e.g. Rossetti, Stella & Zanker, 2015); however, to the best of my knowledge, there is no work yet that specifically addresses the analysis of informational tourism content posted on microblogging services. While review platforms such as yelp.com and tripadvisor.com can help to specifically understand the assessment of certain items by customers, an analysis of microblogging services data could help to investigate high-level industry trends and sentiment. To harness the full power of microblogging data, it is necessary to combine topic models with sentiment

analysis models. Various work about sentiment analysis for Twitter data already exists (e.g. Pak & Paroubek, 2010), whereas topic modeling has largely focused on different domains and larger documents, such as news articles, but has missed out on providing powerful approaches to the analysis of short Twitter text in the tourism domain. This research paper is trying to close exactly this gap and attempts to provide a method to leverage and understand user-generated online information posted on Twitter to realize opportunities to extract value from the vast amount of data related to tourism that can be continuously collected in real-time on this platform.

From a technical point of view, the emphasis will, therefore, be put on topic modeling[2]. The tourism industry will be able to benefit in various ways from accurate and effective topic modeling. Topic modeling of UGC can provide decision support and recommendations to online tourists as well as build a basis for further analytics (Rossetti et al., 2015). Previous research suggests that utilizing retrieved information from Web 2.0 applications and a more effective use of UGC by the tourism industry can point the way to more successful approaches in five key business functions: promotion, product, distribution, communication and management and research (Schmallegger & Carson, 2008). More specifically, the knowledge about the evolution of topics per se could add significant value. Time-sensitive city profiles could be built which could later be matched with user preferences to make user-specific recommendations. Furthermore, an automated extraction of topics and the building of item profiles with scores on each topic is an opportunity to assess the strengths and weaknesses of items and regions as they

[2] A topic model is a statistical machine learning approach that tries to extract coherent thematic information denoted as topics from a large collection of natural language documents (corpora) (Blei, 2012).

are perceived by users (Stella, Cao & Zanker, 2015). Pan et al. (2007), for example, demonstrated the effectiveness and implications of analyzing blog entries in destination marketing. Applying the content analysis approach for 40 blog entries about Charleston, South Carolina, the authors identified that the major strengths of Charleston were its attractions, and that its major weaknesses included weather, infrastructure, and quality of food in fast-service restaurants. Such models could be used for business analytics for management and would allow tourists to compare different service providers and travel destinations. Important aspects and relations from the users' side respecting each of the found topics could be another valuable insight that a topic model can provide and this could uncover unexpected connections between items. Ultimately, in combination with sentiment analysis models and geolocational data[3], powerful machine learning (ML) models that connect emotions and feelings of customers at certain locations with various topics could be built and would create significant value for any agent with a commercial interest in the tourism industry and its trends such as destination marketing organizations or tourism enterprises.

Next to the previously described specific benefits that an effective topic modeling approach could have in the domain of tourism, improvements and extensions in topic modeling in general can be useful for various tasks such as classification, novelty detection, summarization and similarity and relevance judgements (Blei, Jordan, Griffiths, Tennenbaum, 2004). In combination with sentiment analysis models, topic model methods can be exploited to capture different dimensions from UGC that refer to what

[3] This information is readily available on the microblogging service platform Twitter.

users like and what they do not like, and, at the same time, what they assess to be an item's strengths and weaknesses.

Moreover, the automatic topic discovery process can also bring new interesting insights, as it can find connections between words that are typical of the tourism domain. For instance, we can find useful information to understand all the subitems that users usually consider to be important for a specific aspect (Rossetti et al., 2015).

The objective of this research paper is, therefore, to explore different existing approaches based on the topic modeling method with a particular application focus on their usage for Twitter data and microblogging services in the tourism domain and to develop a well-performing model that is able to provide insights about the temporal evolution of topics on such platforms. Whereas most research studies look at one specific topic modeling method, this study implements several different models and compares them to assess their suitability for the type of short texts that is found in Twitter data. Content analysis on Twitter poses unique challenges: posts are short (140 characters or less[4]) with language unlike the standard written English on which many supervised models in ML and natural language processing (NLP) are trained and evaluated. Effectively modeling content on Twitter requires techniques that can readily adapt to the data at hand and require little supervision (Ramage, Dumais, & Liebling, 2010). As a result, four different unsupervised topic modeling techniques will be applied: The Latent Semantic Indexing (LSI) technique, Non-negative Matrix Factorization (NMF), the Latent Dirichlet Allocation (LDA) technique and Hierarchical Dirichlet Processes (HDP). While the major contribution of this work

[4] The character restriction of Twitter has only recently been increased to 280.

should be seen in the application of the topic modeling approach to Twitter data and the potential to apply this approach to various problems in the tourism domain as well as in other domains, this work will specifically investigate the evolution of topics over time in the city of Barcelona. Two major research questions serve as a guideline for this analysis. First, we have seen that insights about the topics prevalent in Twitter could provide significant value for stakeholders interested in the tourism domain. As it is very cumbersome to manually analyze this data, this study will test the role that topic models can play. The first hypothesis belonging to this research question is therefore:

1. It is possible to use ML models to detect reasonable and human interpretable topics from Twitter data related to tourism without human supervision.

Moreover, the second research question will specifically look at the topic environment in the city of Barcelona. First, it could be interesting to know whether there are geographical differences in the mixture of topics. This could help city planners and private stakeholders to assess the strengths and weaknesses of certain areas and adjust destination marketing campaigns for example. The second hypothesis, which will belong to this second research question, therefore is:

2. The topics that will be identified by the topic models show geographical variations and differ from district to district.

Ultimately, it will also be important to be informed about whether the identified topics change over time as this could for example mean that business operations and management should be adjusted according to up-to-date insights and trends. If topics were dynamic over time, this would mean that it is probably helpful to frequently update

the models with new training data and think about real-time analytics solutions. As a result, the third hypothesis, which also pertains to this second research question, is:

3. The topics that are identified by the topic models are stable over time.

Having underlined the importance of obtaining knowledge about the topics prevalent in UGC on social media platforms and, particularly, the microblogging service Twitter for the tourism domain, the following sections will now build towards the implementation and evaluation of the different empirical topic modeling approaches for analyzing the temporal evolution of Twitter data in Barcelona. The city of Barcelona is used as an exemplary case since the study was conducted in Barcelona; and, Barcelona is among the most touristic and most visited cities in Europe. However, the main contribution of this work should be seen in the approach to the problem rather than in the specific case of Barcelona and the dataset at hand. Obviously, this approach could interchangeably be used for other cities as well.

Section 2.1 will start with a short overview of topic modeling to understand the idea behind topic models and their application to text data. Section 2.2 will introduce the research design. Section 2.3 will provide a description of the dataset. Section 2.4 will continue with an outline of the conducted procedure from the design of the study to the implementation of the empirical models to reach the research objective and test the stated hypotheses. Section 2.5 will introduce the actual analysis and describe the specific implementation of the empirical models. Section 3 will describe the results of the empirical models applied to the timestamped Twitter data and Section 4 will explain their relevance in the light of the hypotheses formulated. In Section 5 the study will be concluded through three different

subsections. Section 5.1 will underline the contribution that this study may have to the research field of tourism and to academic scholars in general and how it might help practitioners. Section 5.2 will discuss the limitations of this paper and evaluate how future research could overcome these limitations. Section 5.3 will elaborate on the current state-of-the-art in the research field of social media data analytics in tourism and will propose how a more useful set of conclusions for practitioners can be achieved.

2 Method

2.1 Topic Modeling

As described in the introduction, understanding the informational content of natural language can be of huge value in many domains, including the tourism domain. However, it is not straightforward how exactly a representation of informational content of text should look like. Natural language text has a rich structure: Words on their own can have different meanings or can be used in different ways and contexts by different authors. Words are pieced together in syntactically correct ways to express meaning and oftentimes only adopt their actual semantic meaning when combined with other words into an expression. Many applications in ML and NLP, therefore, cannot use simple word counts but require more advanced mathematical ways to represent or summarize text to conduct computations or extract the most relevant information from a collection of documents. Inferring meaning from text has occupied much of computational linguistics and NLP research through the years; and, different proposals to capture the informational content in the form of topic models have been developed. The objective of topic models is to summarize natural language documents based on coherent thematic information denoted as topics (Rossetti et al., 2015). Informally, a topic should capture “what a document is

about". Hidden semantic structures in natural language have to be identified and taken into consideration to effectively reflect and summarize informational content.

While it is not the objective of this paper to explain all approaches to this challenge in detail, a quick overview of the applied models can help to understand their usefulness in the specific application to Twitter data.

In recent years, we have seen an explosion in topic modeling research and the beginnings of applications of the models and techniques to additional problems and fields. It is expected that these trends will continue in the coming years as approaches that model latent information in data have applicability in a wide range of fields. Different topic modeling techniques have evolved to automatically exploit semantic structures in natural language corpora via the application of mathemical methods.

LSI was the first such technique that was applied to extract topics from a natural language corpus.[5] The LSI technique takes advantage of implicit higher-order (latent) semantic structure in natural language by using singular-value decomposition (SVD) to factorize the term document matrix or preferably the term frequency-inverse document frequency (TFIDF) matrix (Salton & Buckley, 1988)[6] of the text corpus. The idea behind this approach is to represent documents not by the terms as was the case in conventional vector-based methods but by the underlying (hidden, latent) concepts referred to by the terms (Papadimitriou, Raghavan, Tamaki & Vempala, 2000). This allows to identify term

[5] It was originally developed as a new approach to information retrieval to overcome the problem of existing retrieval techniques that tried to simply match words of queries with words of documents without considering conceptual content, thus producing very poor precision and recall measurements (Deerwester, Dumais, Furnas, Landauer & Harshman, 1990; Berry, Dumais & O'Brian, 1995).
[6] TFIDF is used to avoid that large frequencies dominate the squared error deviation used in SVD.

associations prevalent in the text corpus which previously applied methods were unable to detect (Manning, Prabhakar & Schütze, 2008).[7] As a result, a “semantic” space can be constructed, wherein terms and documents that are closely associated are placed near one another. The emerging patterns can then be used to detect the latent components/ topics. Nonetheless, it has to be noted that LSI makes no attempt to interpret the underlying factors nor to “rotate” them to a meaningful orientation but merely tries to apply dimensionality reduction and represent terms and documents in a more reliable lower dimensionality space that is able to reflect term associations (Deerwester et al., 1990)[8]. Although LSI has been applied with remarkable success in different domains, it has a number of deficits and the model performance highly depends on various characteristics of the corpus, mainly due to its unsatisfactory statistical foundation. For example, if the documents are not strongly correlated enough, LSI will produce noise instead of exploiting and bringing out the underlying structure of the corpus. Another major problem with LSI is that the basis vectors have both positive and negative components, and the data are represented as linear combinations of these vectors with positive and negative coefficients. In the application of NLP, however, the negative components contradict physical realities as term frequencies in text mining are non-negative (Pauca, Shahnaz, Berry, & Plemmons, 2004). Also, the meaning of the semantic features cannot be captured intuitively, and the scope of their meaning may be obscure, which is why LSI-related

[7] Through SVD, LSI can for example detect that there is an association between the words “automobile” and “car” by identifying correlations/ co-occurrences between these words in the documents of the corpus.
[8] The rationale behind the increased reliability of a lower dimensional space is that documents which share frequently co-occurring terms will have a similar representation in the latent space, even if they have no terms in common (Hofmann, 2017).

methods of document summarization may fail to extract meaningful sentences (Lee & Seung, 1999; Zha, 2002).

In order to account for this deficit, NMF was developed as a vector space method to obtain a representation of data using non-negativity constraints. These constraints can lead to a parts-based representation because they allow only additive, not subtractive, combinations of the original data. This is in contrast to techniques for finding a reduced dimensional representation based on singular value decomposition-type methods such as LSI (Jolliffe, 2011). Xu, Liu & Gong (2003) demonstrated that NMF-based indexing outperforms LSI for document clustering on a few benchmark collections. Lee, Park, Ahn & Kim (2009) also found that the method selects more meaningful sentences for generic document summarization than those selected using LSI.

In addition to the deterministic approaches LSI and NMF, several probabilistic topic models for documents, most notably the LDA technique, which consists of probabilistic document modeling in which a full generative model for topics and text is described and topics are expressed as hidden random variables, were developed[9]. Unlike the LSI model, the LDA model was developed with a mind to applications beyond information retrieval and uses a three-level hierarchical Bayesian Model, in which each item of a collection is modeled as a finite mixture over an underlying set of topic probabilities and each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities (Blei &

[9] The generative model of a generic document d using the LDA technique consists of the following steps:

- a topic distribution h_d is randomly generated;
- for each word position in d:
    - a topic k is extracted from h_d;
    - a word w is selected with a given probability from topic k (Blei et al., 2003).

Ng, 2003). More specifically, the LDA technique models documents as Dirichlet mixtures (DM) of a fixed number of topics – chosen as a parameter of the model by the user – which are in turn DMs of words. This generates a flat, soft probabilistic clustering of terms into topics and documents into topics.

Another major milestone in the development of topic modeling was set with the introduction of HDP (Teh, Jordan, Beal & Blei, 2004). HDP is a nonparametric Bayesian model (similar in structure to the parametric LDA) for clustering problems involving multiple groups of data. It can be seen as an extension of LDA, designed to address the case where the number of mixture components (the number of "topics" in document-modeling terms) is not known a priori. HDP models topics as mixtures of words, much like LDA, but rather than documents being mixtures of a fixed number of topics, the number of topics is generated by a Dirichlet process, resulting in the number of topics being a random variable as well. The "hierarchical" portion of the name thus refers to another level being added to the generative model (the Dirichlet process producing the number of topics). Unlike its finite counterpart, LDA, which requires some method of model selection, the HDP topic model infers the number of topics from the data. A specification in advance is therefore not necessary. In various applications, the unboundedness of the number of topics can bring a boost in performance. For example, Teh et al. (2004) showed the effective and superior performance of the HDP over previous models on three text corpora.

Having described the theoretical background for the 4 used models, LSI, NMF, LDA and HDP, the following sections will now take a closer look at the empirical study of Twitter text in Barcelona and will explore how the models perform on this corpus.

2.2 Research Design

In line with the nature of the research questions and the typical implementation of a NLP or topic modeling task, this research study can be classified as an observational, descriptive-interpretative, quantitative research study. To ensure an effective approach to the research questions and the hypotheses stated, the research has been separated into two major phases: planning and implementation. Since the dataset has already been provided by the “Operations, Innovation and Data Sciences” faculty at ESADE Business School, it was not necessary to develop a strategy for data collection and sampling methods. The planning phase therefore mainly comprised the construction of a ML pipeline describing all individual steps of the project as well as an adequate pre-selection of models that would be tested in the implementation phase. This phase was thus dominated by a comprehensive literature review of topic modeling approaches and their specific application to Twitter data and by an in-depth study of how an analysis of Twitter data could provide the tourism domain with useful insights. It was conducted between April 2018 and June 2018.

The objective of the implementation phase was then to realize the developed ML pipeline, i.e. clean and prepare the data for the models (including running an exploratory analysis of the data), to train and tune the models, to evaluate the different modeling approaches and select the best-performing model, to apply the model to test the hypotheses and to assess and interpret the results of the model (see Figure 1). This phase covered the time span between July 2018 and September 2018.

Figure 1: Machine learning project pipeline

2.3 Description of the Dataset

The dataset used to answer the research questions consisted of a subsample representing the stream of tourism-related Twitter data in the city of Barcelona. More precisely, it contained Instagram reposts to Twitter in Barcelona made during the period of time between June 11, 2017 and December 4, 2017. This dataset was collected through the Twitter API within the scope of a bigger research project of the "Operations, Innovation and Data Sciences" faculty by ESADE Business School. Within the context of this research study it served as a training corpus for the different topic modeling approaches introduced in this paper as well as a collection of test documents to conduct the final analysis using these models in order to validate the hypotheses.

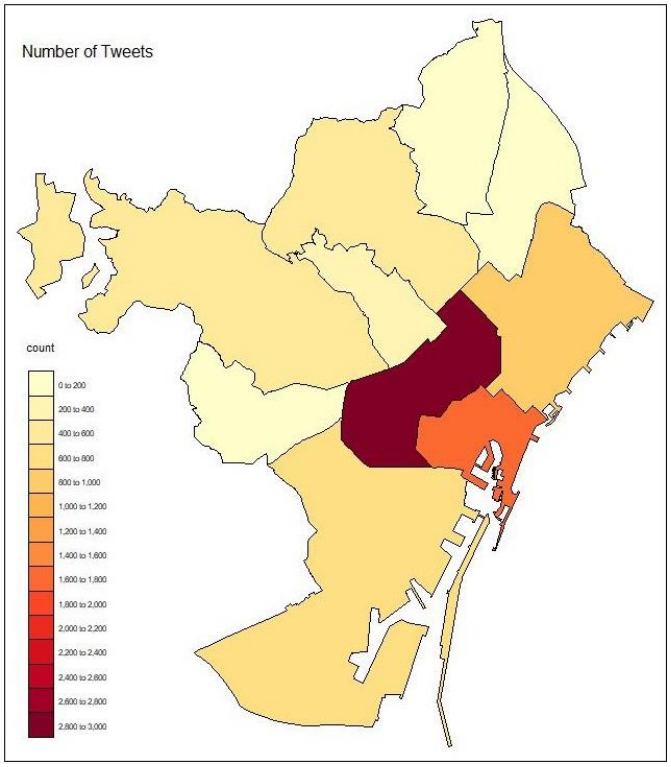
In total, the dataset contained 23778 tweets and had 26 columns, including, among others, variables such as the tweet text, the tweet location (district, neighborhood and coordinates), the tweet time and the language of the tweet. After filtering for English language 7633 tweets remained in the dataset. In addition, the dataset was enriched by spatial data of Barcelona to be able to plot the analysis results in a map of Barcelona. All tweets with a timestamp pertaining to December were only used for training the models but were excluded from the dynamic analysis over time due to the small amount of data available in this month. Some further statistics of the dataset and, particularly, the

characteristics of the tweets after preprocessing, are summarized on a monthly basis in Table 1. A geographical distribution of tweets according to districts is shown in Figure 2.

Table 1: Some statistics of the Twitter dataset after preprocessing

Month	Tweets	Users	Number of Characters	Number of Words	Vocabulary (Unique Words)
June	1270	875	91700	13426	4691
July	1804	1203	129426	18927	6007
August	1187	827	87876	12578	4416
September	1404	917	102928	15347	5083
October	1146	789	86102	12436	4707
November	787	522	60487	8855	3783
December	35	29	2677	380	248

Figure 2: Number of tweets in Barcelona after preprocessing

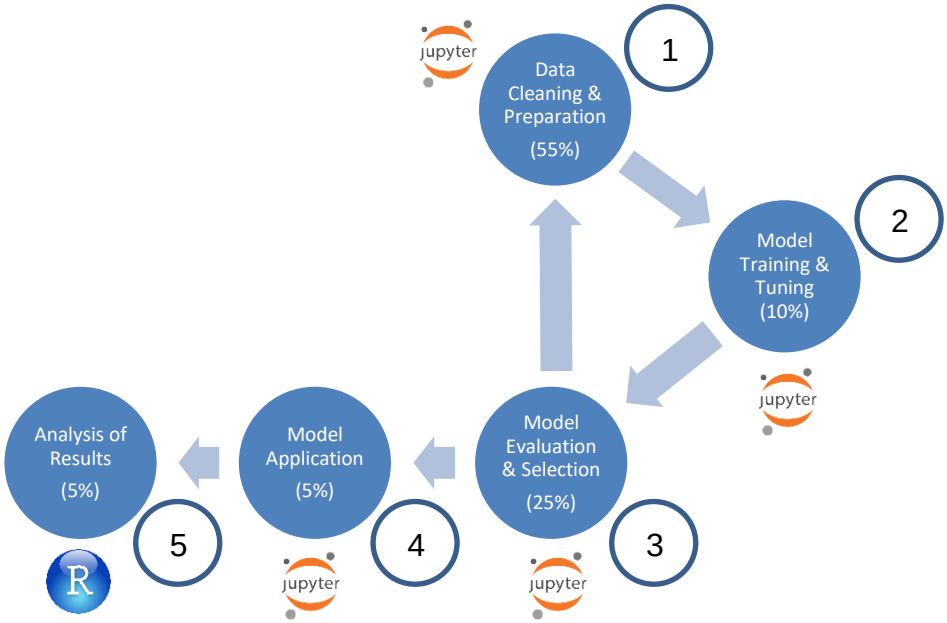


2.4 Procedure

As described in the previous section, a ML pipeline with 5 major subtasks was developed. All of these 5 subtasks and the specific procedure from data cleaning and preprocessing to the analysis of the model results are explored in more detail in this section.

In accordance with usual ML pipelines, the project was not conducted in a waterfall manner, completing one stage after the other, but was, rather, approached as an iterative process with multiple iterations throughout the pipeline. Accumulated, the data preprocessing and cleaning subtask accounted for around 55% of the project workload, the training of the models and model evaluation and selection for around 35% of the workload, and the application of the models to the dataset and analysis of the results for about 10% of the workload. Subtasks 1-4 were implemented in the open source web application Jupyter Notebooks with Python while subtask 5 was conducted with the help of R Studio (see Figure 3).

Figure 3: Distribution of workload and iterative machine learning process



The first part of the data cleaning and preparation subtask (1) was to clean the tweet text. This included the removal of links, the removal of emoticons[10], the removal of punctuation

[10] Emoticons can be useful for sentiment analysis but generate noise and should be removed for topic analysis purposes.

and special characters, such as the removal of certain garbage patterns of characters which were detected after a manual inspection of all tweets in the dataset. Next, the whole twitter text had to be transformed to a format that is compatible with the Python topic modeling packages which were used in the model training & tuning subtask (2), namely gensim and scikit-learn. As both packages require inputs in different data structures, two different preprocessing algorithms had to be implemented.

For the gensim models the whole corpus of Twitter text was transformed into lowercase and then tokenized using white spaces. In addition, numeric tokens and tokens consisting of only a single character were removed and all tokens were lemmatized. The lemmatized tokens were then used as input features for the model training[11]. Bigrams and trigrams were computed and used as additional input features. However, it turned out that the inclusion of bigrams and trigrams severely worsened the model quality, which is why this preprocessing step was excluded in later iterations. Moreover, the English stopwords of the nltk package as well as manually identified stopwords after an inspection of the corpus were removed. In a last step, all tokens that appeared less than two times or in more than 50% of the documents were removed.

For the scikit-learn models, the whole corpus was transformed into lowercase again and stopwords were removed manually from the Twitter text. All the other preprocessing steps were automatically applied through the scikit-learn TFIDF-vectorizer[12].

Furthermore, three different document pooling methods were applied to the corpus:

[11] TFIDF features were also tested instead of the Bag-of-Words (BOW) input but the output was far worse which is why the ultimate model training was based on BOW features.
[12] The NMF model works much better with a TFIDF instead of a BOW input.

1. No Pooling (Each tweet is treated as one document in the modeling process)
2. User Pooling (All the tweets that were posted by the same user combined are treated as one document)
3. Hashtag Pooling (For each hashtag all the tweets that contained this hashtag are combined and treated as one document)

The reason to differentiate between three different pooling methods is that each of the methods has its own advantages and disadvantages; and, some algorithms work better with one method, while others may work better with another method. Hashtag pooling, for example, was shown to work very well with LDA (Mehrotra, Sanner, Buntine, & Xie, 2013) while NMF usually delivers good results without pooling documents (Xu et al., 2003).

In the second subtask, model training and tuning (2), the documents provided by the different pooling methods were used to train different topic models. While most of the research previously conducted in the field of topic modeling generally focuses on one specific model, this work follows a different approach and tries to compare different models that are widely established to assess how suitable they are for the analysis of Twitter data. As a consequence, LSI, LDA, HDP and NMF were all applied to the Twitter corpus resulting from the dataset. Whereas LSI, LDA and HDP were tested with all three different pooling methods, NMF was only used with the no pooling approach, as it is shown to work very well with short documents and does not need a lot of semantic structure to work (Xu et al., 2003). LDA and HDP are probabilistic models and were, therefore, run multiple times. Only the best-performing model was saved and compared with the other models. LSI and NMF, in contrast, are deterministic, which is why they were only run once.

All the models were tuned by playing around with different hyperparameters, especially, the number of topics, if applicable[13].

In the third subtask, model evaluation and selection (3), all trained models were compared. This comparison was based on both quantitative and qualitative measures. The LSI, LDA and HDP models were evaluated using the c_v coherence score implemented in the gensim package. The NMF models were evaluated using the TC-W2V coherence score. In addition, all top terms of each topic of all the models were inspected by hand and qualitatively assessed on their human interpretability by a committee of 2 persons. The pyLDAvis package was used to visualize the LDA topic models, displaying the topic position in a principal component axis graph and showing the top 30 most salient terms for each topic. Nonetheless, it has to be stated that it is difficult to compare different models, because there is no principled method for evaluation of a topic model. Often evaluation is only done through hand-inspection of topics and comparison with those from a simpler model to show that a new model better captures some property of text documents, because quantitative measures such as coherence scores or the perplexity measure often fail to provide human interpretable topics. While it is true that for certain tasks human interpretability is not necessary[14], this study emphasizes the importance of interpretability to extract useful insights. It was, therefore, decided to use a manual inspection and qualitative assessment of the topics as major factor for the model selection.

As described in Figure 3, the model evaluation often led to an iterative process by generating new insights that had to be accounted for in the data cleaning and

[13] The HDP model automatically determines the optimal number of topics.
[14] This could for example be the case if the outputs of the topic models serve as input features for another ML model.

preprocessing subtask, which then, in turn, resulted in a new model training and evaluation phase. This iterative process combined with a tuning of the hyperparameters was one of the major challenges of the research project.

In the fourth subtask, model application (4), the objective was to apply the model to the dataset in a way such that the research hypotheses could be tested. Therefore, the dataset was divided into two subsets. In a first subset, the tweets were ordered according to their geographic origin only and all the tweets in one district of Barcelona were treated as a test document (from now on referred to as test collection 1). This collection of test documents would serve to answer the second hypothesis[15]. In a second subset, the tweets were ordered according to their geographic region and according to their timestamp, or, more specifically, the month in which the tweet was posted (from now on referred to as test collection 2). This collection of test documents would serve to answer the third hypothesis[16]. The idea was to use the best-performing topic model to find out the topic mixture and topic scores for each test document which would then reflect the topic mixture in a district (test collection 1) or in a district during one specific month (test collection 2). By looking at different months, a dynamic analysis could be conducted to answer the third hypothesis.

In the last subtask, analysis of results (5), the topic scores are communicated through a map of the city of Barcelona and compared with each other to validate the hypotheses.

[15] Recall Hypothesis 2: “2. The topics that will be identified by the topic models show geographical variations and differ from district to district.
[16] Recall Hypothesis 3: “3. The topics that are identified by the topic models are stable over time.”

2.5 Data Analysis

After having outlined the procedure, this section will explain the thoughts accompanying each step and will go into further detail regarding the analysis of the data and models. While this section will focus on the most relevant analysis, a comprehensive overview can be found in the Appendix, where the complete code for the project including explanatory comments is provided.

As mentioned in previous sections, the Twitter corpus was prepared according to three different pooling methods. Summary statistics for each method can be found in Table 2.

Table 2: Pooling methods summary statistics

Pooling Method	Documents	Total Terms	Average Terms per Document
No Pooling	7633	82165	10.76
User Pooling	4424	82028	18.54
Hashtag Pooling	6040	150793	24.97

In the model and tuning phase (2), the following models were trained: For the probabilistic LDA variants ten iterations with five models each were processed, i.e. 50 models in total for each pooling method, respectively[17]. One Iteration refers in this case to the training of five models; as it included varying the topic parameter in a range between four and eight, while holding all other parameters constant. For each iteration the model with the highest c_v coherence score was saved and compared to the models of other iterations. In each iteration the hyperparameters “alpha”, “eta”, “iterations” and “passes” were tuned[18]. The iteration with the best performing model for each pooling method is displayed in Figure 4 (No Pooling), Figure 5 (User Pooling) and Figure 6 (Hashtag Pooling). For the No Pooling

[17] 50 no pooling models, 50 user pooling models and 50 hashtag pooling models were trained.
[18] For a detailed explanation of the hyperparameters compare the official gensim documentation under https://radimrehurek.com/gensim/models/ldamodel.html.

and Hashtag Pooling Method, the best-performing model turned out to have a number of topics of seven, while for the User Pooling Method the best-performing model had six topics.

Figure 4: Coherence score comparison of the best-performing No Pooling LDA iteration

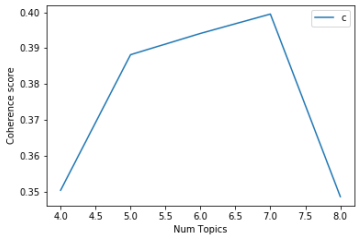


Figure 5: Coherence score comparison of the best-performing User Pooling LDA iteration

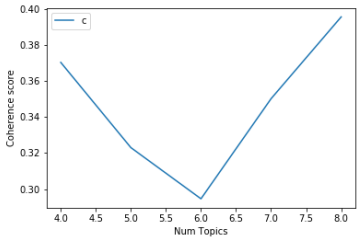
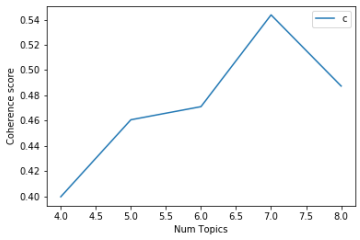


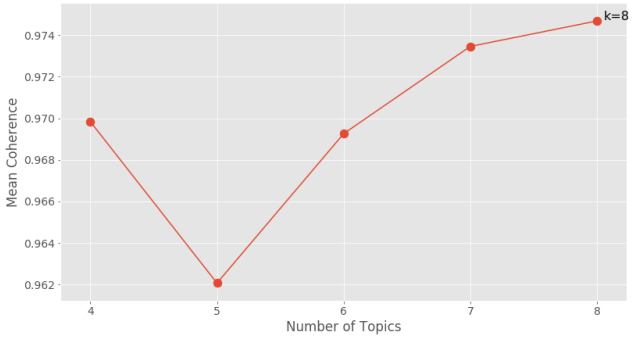
Figure 6: Coherence score comparison of the best-performing Hashtag Pooling LDA iteration



For the deterministic NMF model one iteration with five models was trained, again varying the topic number parameter. It has to be emphasized that there is no implementation of the NMF model yet in the gensim package, which is why the scikit-learn package was used to build this model. Due to differences between the packages, a different coherence model, the TC-W2V coherence model, was used to find the best-performing model of each

iteration. The result with the best TC-W2V coherence was saved for comparison with other topic modeling methods. The best-performing model had a topic number of eight in the case of the NMF model (see Figure 7).

Figure 7: Coherence score comparison of the NMF No Pooling models



The procedure for training the HDP models was to run ten iterations of a single model as topics are automatically determined by this model. Again, for each pooling method, the best-performing model was saved and compared to the other models.

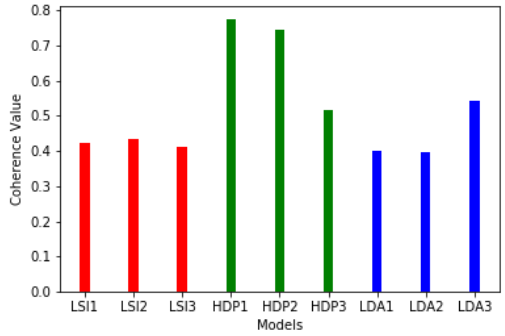
For the LSI model one iteration with five models varying the number of topics was trained. Due to clear underperformance compared to the other models, the next nine iterations were run with a fixed number of topics, which was for each pooling method set to the optimal number of topics determined by the LDA models.

After comparing all the models the following result was obtained: While various models can indeed extract meaningful topics from the Twitter dataset (even given the small number of 7633 tweets that remained after the preprocessing step), the experiment conducted in this study shows that the Hashtag Pooling LDA model reaches more human interpretable topics and higher c_v coherence scores than the LSI as well as the No Pooling and User Pooling LDA models. This is in line with previous research that found

that standard No Pooling LDA does not work well with Twitter because tweets are short (Zhao et al., 2011).

The HDP models obtain even higher coherence scores; but, this is due to the automatic determination of the number of topics, which leads to a higher number of topics than the limit of eight to which the other models were restrained[19]. Upon manual inspection of the topics, the HDP topics of all three pooling methods seem too granular for this small dataset, which is why it may not be the best model despite its high coherence scores (see Figure 8).

Figure 8: Coherence score comparison of LSI, HDP and LDA models



Since quantitative measures tend to be unreliable for the assessment of topic models and different coherence scores were implemented for the scikit-learn and gensim packages, the final comparison between the NMF model and the LSI, HDP and LDA models was primarily based on a human inspection of the topics. However, evaluation of topic models is regularly done through hand-inspection and tends to deliver useful results. Therefore, two human judges were asked to assign a qualitative score to each topic for each model

[19] In fact, if LDA were allowed to have a similar number of topics it would reach similar coherence scores.

according to the following guidelines based on the top 10 topic words and their average was used as the score for each topic and model:

- Score of 1: meaningful and coherent
- Score of 0.5: containing multiple topics or noisy words
- Score of 0: making no sense

The results of this qualitative assessment are displayed together with their annotation agreement information in Table 3. Cohen's Kappa[20] was used to measure inter-rater agreement.

Table 3: Qualitative assessment of topic models

Model	Avg. Score	Agreement between Judges	Cohen's Kappa
LDA No Pooling	0.74	80.6%	0.71
LDA User Pooling	0.72	82.0%	0.73
LDA Hashtag Pooling	0.81	92.0%	0.88
LSI No Pooling	0.31	78.6%	0.68
LSI User Pooling	0.36	82.9%	0.74
LSI Hashtag Pooling	0.40	68.6%	0.53
HDP No Pooling	0.71	82.2%	0.73
HDP User Pooling	0.69	79.9%	0.70
HDP Hashtag Pooling	0.58	81.4%	0.72
NMF No Pooling	0.65	82.5%	0.74

[20] Cohen's Kappa takes on a value of 1 for perfect agreement and 0 for pure randomness.

We can see that the Hashtag Pooling LDA model outperforms the other models, giving even more meaningful top topic words. The Hashtag Pooling LDA model is, therefore, selected as the best-performing topic model and is used to test the hypotheses.

With the help of this model, each of the documents in test collection 1 and test collection 2 will be associated with a mixture of Twitter topics. The results of this process are presented in the next section.

3 Results

3.1 Results concerning Hypothesis 1

The selected model, the Hashtag Pooling LDA model with seven topics, identified the topics that are presented in Table 4 within the city of Barcelona. These topics were manually labelled after inspection of some of the top feature terms comprising the topic. The ranking was automatically determined by the pyLDAvis package. Figure 11 shows a wordcloud with the top 10 words for each topic.

Table 4: Topics of Barcelona

Topic Number	Top Coherence Terms	Label	Topic Ranking (Importance)
0	workout, fit, meditation, healthy, video	Sports, Health & Image	7
1	yoga, selfie, contemporaryart, yummy, brianeno	Lifestyle & Culture	5
2	night, olgod (a beer bar), cocktail, beer, raval	Nightlife	6
3	graffiti, streetart, arte urbano, massive, streetphotography	Streetart	3
4	sagrada familia, gaudi, architecture, travel, church	Sightseeing	1
5	beach, friends, summer, smile, sun	Summer, Sun & Friends	2
6	yum, home, place, call, tapas	Everyday Life	4

Figure 9: Topic wordclouds

We can see that the LDA Hashtag Pooling model was in fact able to detect reasonable and human interpretable topics and can, thus, confirm the first hypothesis.

3.2 Results concerning Hypothesis 2

Conducting the geographical analysis of these topics resulted in the map for the city of Barcelona displayed in Figure 12. Dark red areas represent high topic scores while light yellow areas represent low topic scores. The minimum and maximum topic scores as well as the average topic scores over all districts for each topic are shown in Table 5. We can already see that “Sightseeing” is, as expected, by far the most relevant topic followed by “Streetart”, “Everyday Life” and “Summer, Sun & Friends”. Regarding the geographical distribution of topics[21], the topic “Sightseeing” is, as expected, primarily talked about in the district of Eixample where Sagrada Familia is located. The topic “Streetart” is distributed all over the city, but primarily present in the districts Les Corts, Sants-Montjuïc and Sant Martí. The topic “Everyday Life” has its highest score in Horta-Guinardó and the topic “Summer, Sun & Friends” ,as expected, in Ciutat Vella, the district with the beach of

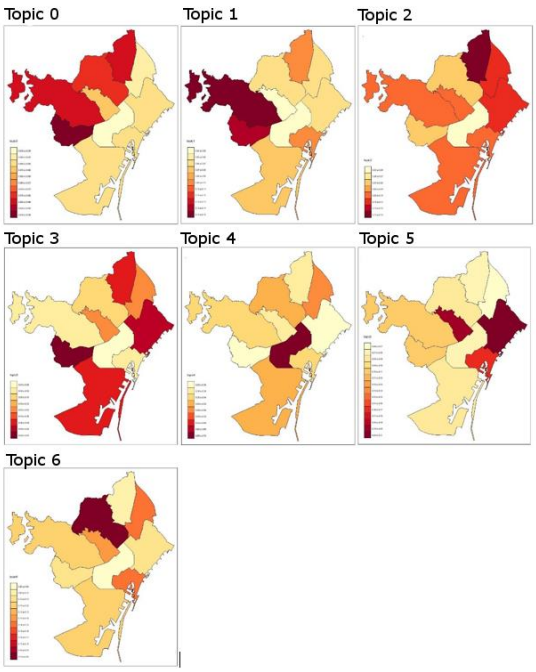
[21] For a map showing the names of all Barcelona districts see the Appendix.

Barceloneta, and Sant Martí. In general, the topic scores very much vary with the district and the tourism activities that each district has to offer. The findings are, therefore, very much in line with hypothesis 2 that the presence of identified topics varies from district to district in the city of Barcelona. While it is not possible to calculate confidence intervals for this finding due to the changing topics of topic models, a testing with models from different iterations confirms that the hypothesis is very likely to hold and certainly cannot be rejected.

Table 5: Topic score summary

Topic Number	Label	Mean Topic Score	Minimum Topic Score (Light Yellow)	Maximum Topic Score (Dark Red)
0	Sports, Health & Image	0.064	0.040	0.095
1	Lifestyle & Culture	0.082	0.040	0.150
2	Nightlife	0.093	0.050	0.130
3	Streetart	0.126	0.040	0.220
4	Sightseeing	0.392	0.250	0.700
5	Summer, Sun & Friends	0.118	0.060	0.210
6	Everyday Life	0.124	0.080	0.200

Figure 10: Topic map of Barcelona



3.3 Results concerning Hypothesis 3

Taking a look at the dynamic analysis over time now, the results displayed in Figure 11 to 17 can be found.

Figure 12: Dynamic analysis: Sports, Health & Image

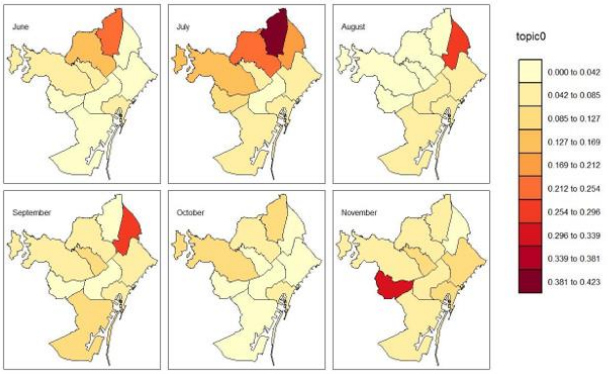


Figure 11: Dynamic analysis: Streetart

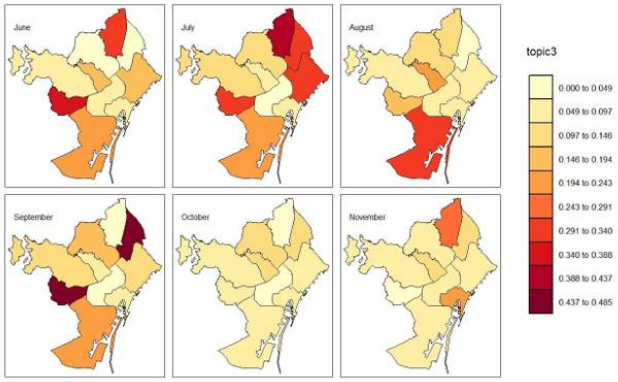


Figure 14: Dynamic analysis: Lifestyle & Culture

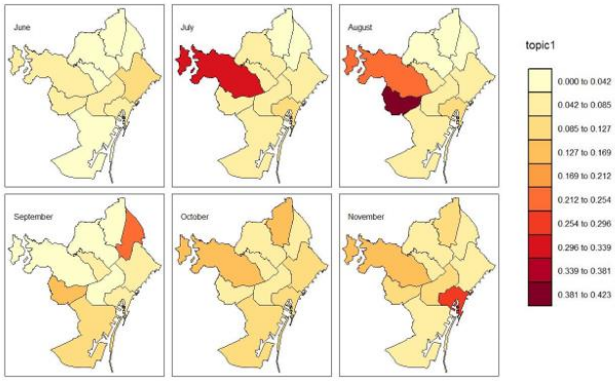


Figure 13: Dynamic analysis: Sightseeing

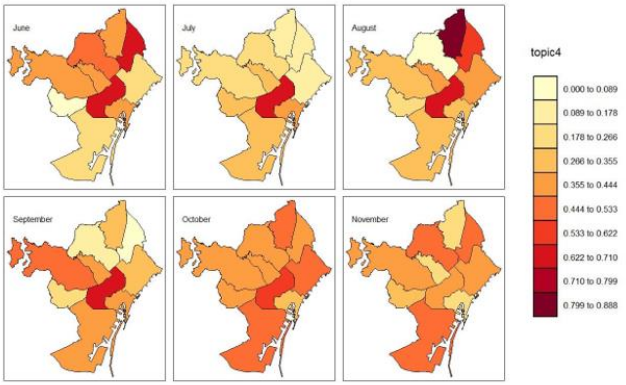


Figure 16: Dynamic analysis: Nightlife

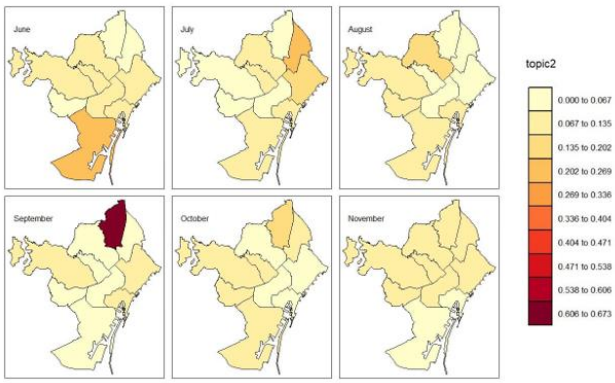


Figure 15: Dynamic analysis: Summer, Sun & Friends

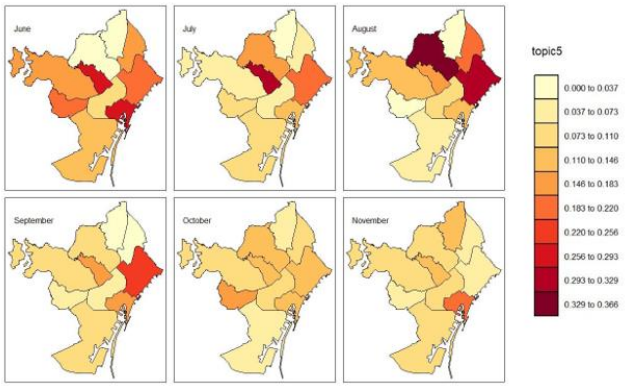
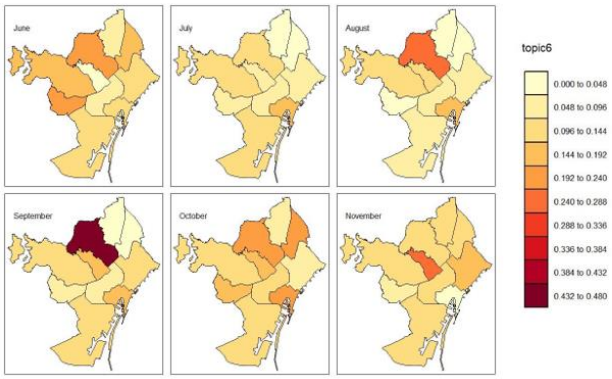


Figure 17: Dynamic analysis: Everyday Life



The first observation that we can make is that the topics are not stable but vary over time, some more than others. The “Sightseeing” topic, for example, is rather stable. The “Summer, Sun & Friends” topic is primarily present during summer time and the “Nightlife” topic is channeled on one month and one district, probably an indicator for a single event or group of people posting multiple tweets about the same activity. The “Everyday Life” topic is regularly located in the Horta-Guinardó district but shows different intensities over time. The “Streetart” topic is also primarily found during the summer months. In general, it seems that topics are dynamic over time. As a result, the third hypothesis, which states that the topics that are identified by the topic models are stable over time, cannot be confirmed. Again looking at different iterations manually to compensate for the lack of confidence intervals, this result seems to be stable.

4 Discussion

Although this research study does not introduce new topic models to address the issues of short text modeling, especially in microblogging environments, in this study, the results shed some light on how research on topic models can be conducted for short text

scenarios. As we have seen in the previous section, three major results regarding the two research questions can be summarized:

1. It is possible to use ML models to detect reasonable and human interpretable topics from Twitter data related to tourism without human supervision.
2. The topics identified by the best-performing model (LDA Hashtag Pooling) for the city of Barcelona show geographical variations and differ from district to district.
3. The identified topics are not stable but change over time.

The result for the first hypothesis is very much in line with previous research that found that topic modeling is indeed a powerful tool not only for the analysis of long text documents but also for short text messages. Multiple experiments also showed that aggregating messages may lead to superior performance (Hong & Davison, 2010). This observation was also made in this study as the Hashtag Pooling and User Pooling models outperformed the No Pooling models. Particularly, pooling according to hashtags has been proven successful in other studies as well (Mehrotra et al., 2013).

Regarding the second and third hypothesis and result, there is, to the best of my knowledge, no previous work that has addressed these issues in the tourism domain. Nonetheless, the results are not very surprising as different geographical locations offer different tourism activities and have different advantages and disadvantages, which also vary over time; and, this is reflected in the topics. In fact, it is exactly the detection of this variability which could make topic models a powerful tool for stakeholders in the travel and tourism domain. Obviously, scholars and practitioners could adapt the approach presented in this study to obtain relevant results for their specific research question or

business objective, for example, by filtering the tweets for certain hashtags and thus getting a more granular analysis of a certain subcategory of the travel and tourism industry.

While this study only looked at the city of Barcelona and does not provide any proof of generalization, it is very likely that other touristic cities will show similar results. In general, a geographical mapping of tweets was shown to provide planners, marketers and researchers with useful information about activities and opinions across time and space in other cities such as London, although not particularly in the tourism domain (Lansley & Longley, 2016).

Moreover, potential extensions of the applied models, such as labeled LDA (Ramage, Dumais, & Liebling, 2010), may provide even better results and could be looked at in future research. Nonetheless, other work in the field also showed that it is hard to develop well-performing extensions and that, oftentimes, the standard LDA model with aggregated documents delivers the best results (Hong & Davison, 2010).

It is, therefore, without a doubt that standard topic models have proven to be a useful tool for discovering latent structures in document collections. However, another issue is that most document collections often come as temporal streams. Thus, several aspects of the latent structure such as the number of topics, the topics' distribution and popularity are time-evolving (Ahmed & Xing, 2010). In order to account for this fact, the topic model method has been extended in several ways for dealing with topics’ evolution over time (Blei and Lafferty, 2006b). Future research could try to apply such extensions to Twitter data and compare the results over time with traditional batch training models.

5 Conclusion

5.1 Main Contributions of this Paper

This paper has developed a dynamic topic analysis method for georeferenced tweets in the city of Barcelona. The presented Twitter sample demonstrates that with a large enough sample of data and robust text cleaning techniques, Hashtag Pooling LDA can overcome the challenge of short tweets and provide reasonable and human interpretable topics.

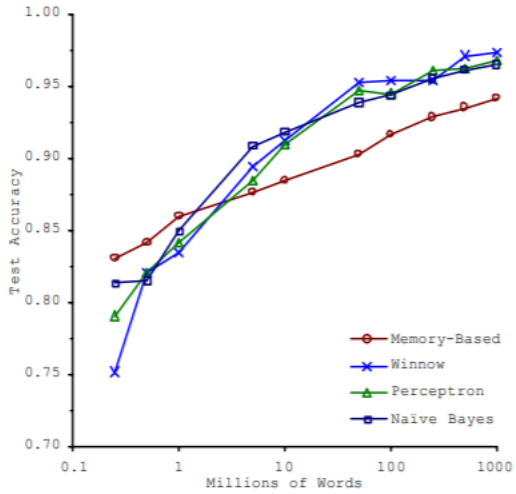
The main contribution of this paper is therefore an end-to-end algorithm from raw Twitter data to a mapping of the dynamic evolution of geographical topics, which can theoretically be applied to multiple domains and problems to analyze high-level market trends and mechanisms. Moreover, in combination with sentiment analysis, such an algorithm provides researchers and practitioners with the opportunity to detect opinions about and strengths and weaknesses of specific items which can be of considerable managerial value. Knowledge about the geography of topics on social media can also contribute to understand the social dynamics of urban areas. This knowledge is usually not readily available from traditional datasets.

5.2 Limitations

With regard to the limitations of this research, one of the major aspects to emphasize is that ML models and, particularly, NLP models stand or fall with the quality and amount of data processed. In their famous paper Banko & Brill (2001) showed that the model quality actually heavily depends on the data fed into the model and is only slightly influenced by the model selection (see Figure 18). Similar to their suggestion that time should rather be spent on corpus development than on algorithm development, a larger and more variable

dataset could probably substantially improve the results of the topic modeling approach conducted in this research. With a number of only 7633 tweets after filtering for English language, the topic models are pretty unstable and easily distorted by outlier tweets. Moreover, a filter for certain hashtags could allow for more granular and thus more useful topics that are not as generic as for example “Sightseeing” or “Summer, Sun & Friends”. In the end, however, the granularity and a potential narrowing down according to certain hashtags should be adapted in line with the research question and the addressed stakeholders. This could be done very easily by a slight modification of the approach suggested in this paper.

Figure 18: The importance of data for model performance



Another limitation that is not directly related to the results and the approach presented in this paper but rather to the technical implementation of it is that the code was executed on a local machine with 8GB RAM and an i7-5500U CPU. The training of the LSI, LDA and HDP models was, therefore, very slow and a lot of time was lost due to limited processing speed. This could be avoided through a multicore implementation of the

algorithm that was used in this research using more powerful machines provided in the cloud.

Yet another limitation of this study is that the implemented models only allow for batch training. However, as we have seen, topics change over time and a dynamic corpus training is therefore necessary to capture new unforeseen trends. As a result, I would recommend that future research in this area focuses on dynamic variants of the models presented in this work. These models should take into account the dynamic environment prevalent in the tourism domain and allow for online training on the spot.

5.3 Prospective

The literature review that was conducted prior to the implementation of the ML pipeline covered both a comprehensive review of the current state-of-the-art of topic models and a detailed investigation of the most pressing research questions in the travel and tourism domain. However, retrospectively, I think focusing more on the particular application of topic models to Twitter data could have been useful as well. As mentioned earlier, there are several extensions of the described models that try to specifically overcome the challenges of tweets which bear a character restriction and are often characterized by slang language that is uncommon for the type of documents for which the models presented in this paper were originally developed. Nonetheless, the character restriction of tweets has only lately been increased from 140 to 280, which might increase the effectiveness of traditional topic models.

6 References

Ahmed, A., & Xing, E. P. (2010). Timeline: A dynamic hierarchical Dirichlet process model for recovering birth/death and evolution of topics in text stream. *UAI'10 Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence* (pp. 20-29). Catalina Island: AUAI.

Akehurst, G. (2009). User generated content: the use of blogs for tourism organisations and tourism consumers. *Service Business, 3(1)*, 51-61.

Aletras, N., & Stevenson, M. (2014). Labelling topics using unsupervised graph-based methods. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (pp. 631-636). Baltimore: Association for Computational Linguistics.

Ballantyne, R., Packer, J., & Axelsen, M. (2009). Trends in tourism research. *Annals of Tourism Research, 36(1)*, 149-152.

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26-33). Association for Computational Linguistics.

Barbier, G., & Liu, H. (2011). Data mining in social media. *Social network data analytics*, 327-352.

Berry, M. W., Dumais, S. T., & W., O. G. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review, 37(4)*, 573-595.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM, 55(4)*, 77-84.

Blei, D. M., & Ng, A. Y. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research, 3(Jan)*, 993-1022.

Blei, D. M., Jordan, M. I., Griffiths, T. L., & Tenenbaum, J. B. (2004). Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 17-24.

Chan, N. L., & Guillet, B. D. (2011). Investigation of social media marketing: how does the hotel industry in Hong Kong perform in marketing on social media websites? *Journal of Travel & Tourism Marketing, 28(4)*, 345-368.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS quarterly*, 1165-1188.

Dahl, D. (2013). Social influence and consumer behavior. *Journal of Consumer Research, 40(2)*, iii-v.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science banner*, 391-407.

Dellarocas, C. (2003). The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Management Science, 29(10)*, 1407-1424.

Dippelreiter, B., Grün, C., Pöttler, M., Seidel, I., Berger, H., Dittenbach, M., & Pesenhofer, A. (2008). Online tourism communities on the path to Web 2.0: an evaluation. *Information technology & tourism, 10(4)*, 329-353.

Fesenmaier, D. R., Wöber, K. W., & Werthner, H. (2006). *Destination recommendation systems: Behavioural foundations and applications.* Wallingford: CAB International.

Gantz, J., & Reinsel, D. (2012). The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. *IDC iView: IDC Analyze the future*, 1-16.

Gimpel, K. (2006). *Modeling topics.* Pittsburgh: Carnegie Mellon University.

Gretzel, U., & Yoo, K. H. (2008). Use and impact of online travel reviews. *Information and communication technologies in tourism*, 35-46.

Gretzel, U., Fersenmaier, D. R., & O'leary, J. T. (2006). The transformation of consumer behavior. *Tourism business frontiers*, 31-40.

Hofmann, T. (1999). Probabilistic latent semantic analysis. *SIGIR '99 Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 50-57). New York: ACM New York.

Hofmann, T. (2017). Probabilistic latent semantic indexing. *ACM SIGIR Forum (Vol. 51, No. 2, pp. 211-218)* (pp. 211-218). Berkeley: ACM.

Hong, L., & Davison, B. D. (2010). Empirical study of topic modeling in twitter. *Proceedings of the first workshop on social media analytics* (pp. 80-88). ACM.

Howison, S., Finger, G., & Hauschka, C. (2015). Insights into the Web presence, online marketing, and the use of social media by tourism operators in Dunedin, New

Zealand. *An International Journal of Tourism and Hospitality Research 26(2)*, 269-283.

Inversini, A., Cantoni, L., & Buhalis, D. (2009). Destinations' information competition and web reputation. *Information technology & tourism, 11(3)*, 221-234.

Jeng, J., & Fesenmaier, D. R. (2002). Conceptualizing the travel decision-making hierarchy: A review of recent developments. *Tourism Analysis, 7(1)*, 15-32.

Jolliffe, I. (2011). *Principal component analysis.* Berlin, Heidelberg: Springer.

Klein, L. R. (1998). Evaluating the potential of interactive media through a new lens: Search versus experience goods. *Journal of Business Research, 41(3)*, 195-203.

Lansley, G., & Longley, P. A. (2016). The geography of Twitter topics in London. *Computers, Environment and Urban Systems 58*, 85-96.

Lau, J. H., Grieser, K., Newman, D., & Baldwin, T. (2011). Automatic labelling of topic models. *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (pp. 1536-1545). Portland: Association for Computational Linguistics.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 788-791.

Lee, J. H., Park, S., Ahn, C. M., & Kim, D. (2009). Automatic generic document summarization based on non-negative matrix factorization. *Information Processing & Management, 45(1)*, 20-34.

Leung, D., Law, R., van Hoof, H., & Buhalis, D. (2013). Social media in tourism and hospitality: A literature review. *Journal of travel & tourism marketing, 30(1-2)*, 3-22.

Li, X., & Wang, Y. (2011). China in the eyes of western travelers as represented in travel blogs. *Journal of Travel & Tourism Marketing, 28(7)*, 689-719.

Litvin, S. W., Goldsmith, R. E., & Pan, B. (2008). Electronic word-of-mouth in hospitality and tourism management. *Tourism Management, 29(3)*, 458-468.

Liu, S. Q. (2005). A theoretic discussion of tourism e-Commerce. *Proceedings of the 7th international conference on Electronic Commerce* (pp. 1-5). New York: ACM.

Magatti, D., Calegari, S., Ciucci, D., & Stella, F. (2009). Automatic Labeling of Topics. *ISDA '09 Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications* (pp. 1227-1232). Washington: IEEE Computer Society.

Manning, C. D., Prabhakar, R., & Schütze, H. (2008). *Introduction to information retrieval.* New York: Cambridge University Press New York.

María Munar, A. (2011). Tourist-created content: rethinking destination branding. *International Journal of Culture, Tourism and Hospitality Research, 5(3)*, 291-305.

Mehrotra, R., Sanner, S., Buntine, W., & Xie, L. (2013). Improving lda topic models for microblogs via tweet pooling and automatic labeling. *36th International ACM SIGIR conference on Research and development in information retrieval* (pp. 889-892). ACM.

Nelson, P. (1970). Information and consumer behavior. *Journal of political economy, 78(2)*, 311-329.

Noone, B. M., McGuire, K. A., & Rohlfs, K. V. (2011). Social media meets hotel revenue management: Opportunities, issues and unanswered questions. *Journal of Revenue and Pricing Management, 10(4)*, 293-305.

O'Connor, P. (2008). User-generated content and travel: A case study on Tripadvisor.com. *Information and Communication Technologies in Tourism*, 47-58.

Paisley, J., Wang, C., Blei, D. M., & Jordan, M. I. (2014). Nested hierarchical Dirichlet processes. *Journal of Pattern Analysis and Machine Intelligence, 37(2)*, 256-270.

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. *LREc (Vol. 10, No. 2010)*, 1320-1326.

Pan, B., MacLaurin, T., & Crotts, J. C. (2007). Travel blogs and the implications for destination marketing. *Journal of Travel Research, 46(1)*, 35-45.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval, 2(1-2)*, 1-135.

Pantelidis, I. S. (2010). Electronic meal experience: A content analysis of online restaurant comments. *Cornell Hospitality Quarterly, 51(4)*, 483-491.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences, 61*, 217-235.

Pauca, V. P., Shahnaz, F., Berry, M. W., & Plemmons, R. J. (2004). Text mining using non-negative matrix factorizations. *Society for Industrial and Applied Mathematics*, (pp. 452-456).

Qiang, Y., Law, R., Gu, B., & Chen, W. (2011). The influence of user-generated content on traveler behavior: An empirical investigation on the effects of e-word-of-mouth to hotel online bookings. *Computers in Human Behavior, 27(2)*, 634-639.

Ramage, D., Dumais, S., & Liebling, D. (2010). Characterizing Microblogs with Topic Models. *ICWSM 10(1)*, (p. 16).

Rossetti, M., Stella, F., & Zanker, M. (2015). Analyzing user reviews in tourism with topic models. *Information Technology & Tourism, 16(1)*, 5-21.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24(5)*, 513-523.

Schmallegger, D., & Carson, D. (2008). Blogs in tourism: Changing approaches to information exchange. *Journal of Vacation Marketing, 14(2)*, 99-110.

Sotiriadis, M., & Van Zyl, C. (2013). Electronic word-of-mouth and online reviews in tourism services: the use of twitter by tourists. *Electronic Commerce Research, 13(1)*, 103-124.

Stella, F., Cao, L., & Zanker, M. (2015). Analysing user reviews in tourism with topic models. *Information Technology & Tourism, 16(1)*, 5-21.

Stepchenkova, S., & Morrison, A. M. (2006). The destination image of Russia: From the online induced perspective. *Tourism Management*, 943-956.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2004). Sharing clusters among related groups: Hierarchical Dirichlet processes. *NIPS'04 Proceedings of the 17th International Conference on Neural Information Processing Systems* (pp. 1385-1392). Vancouver: MIT Press Cambridge.

Thevenot, G. (2007). Blogging as a social media. *Tourism and Hospitality Review, 7(3-4)*, 287-289.

Walker, S. (2014). Big data: A revolution that will transform how we live, work, and think. *International Journal of Advertising, 33(1)*, 181-183.

Werthner, H., & Klein, S. (1999). *Information technology and tourism: a challenging relationship.* Wien: Springer-Verlag Wien.

Xiang, Z., & Gretzel, U. (2010). Role of social media in online travel information search. *Tourism Management*, 179-188.

Xiang, Z., Magnini, V. P., & Fesenmaier, D. R. (2015). Information technology and consumer behavior in travel and tourism: Insights from travel plannung using the internet. *Journal of Retailing and Consumer Services, 22*, 244-249.

Xiang, Z., Wöber, K., & Fesenmaier, D. R. (2008). Representation of the online tourism domain in search engines. *Journal of Travel Research, 47(2)*, 137-150.

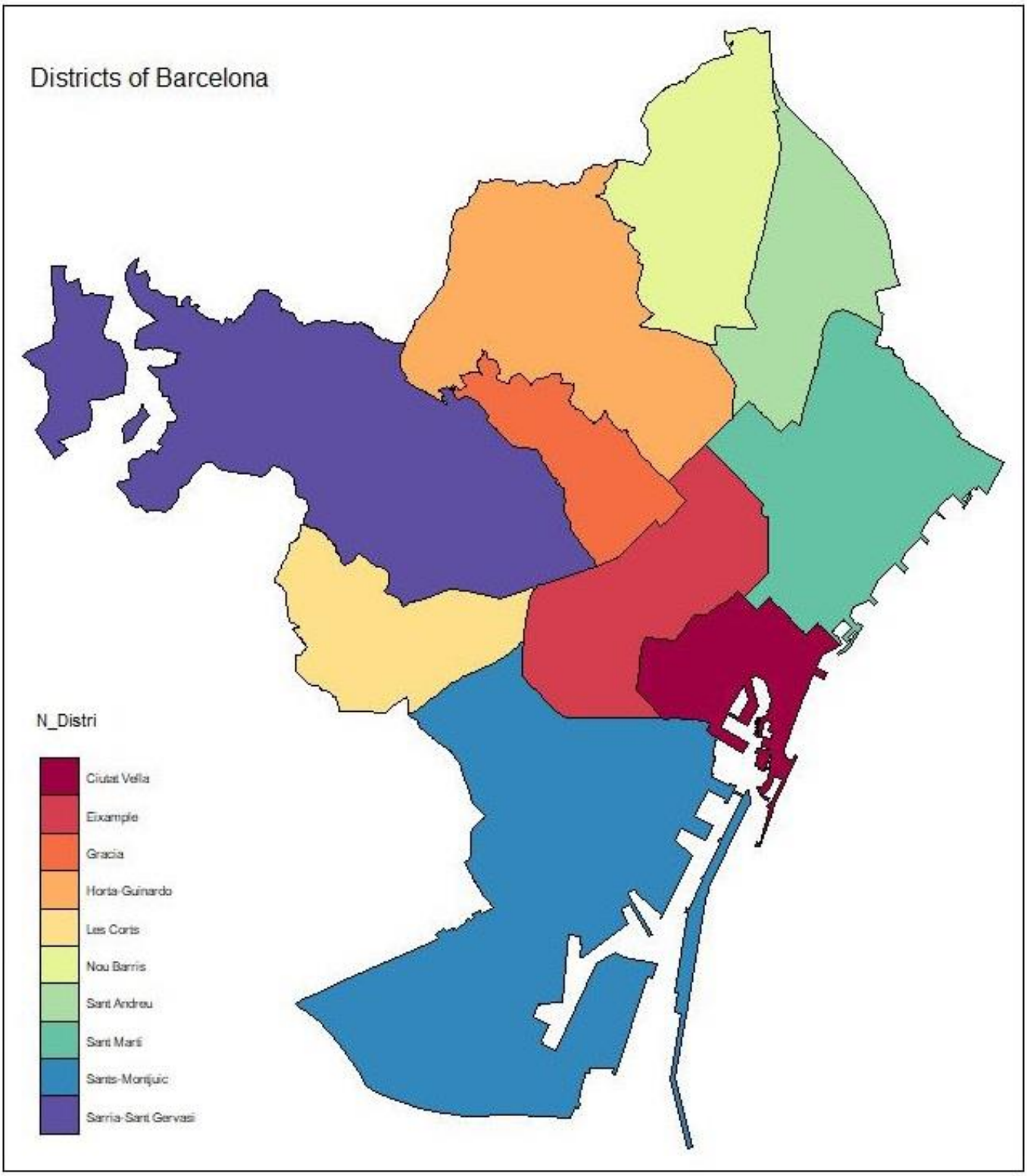
Xu, W., Liu, X., & Gong, Y. (2003). Document-clustering based on non-negative matrix factorization. *SIGIR'03*, (pp. 267-273). Toronto.

Zha, H. (2002). Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. *SIGIR'02*, (pp. 113-120). Tampere.

Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., & Li, X. (2011). Comparing twitter and traditional media using topic models. *European conference on information retrieval* (pp. 338-349). Heidelberg: Springer.

Appendix

Appendix 1: Map of Barcelona districts



Code

Jupyter Notebook 1: Preprocessing

References

In [8]:
```
# https://radimrehurek.com/gensim/tut1.html
# https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/
# https://docs.python.org/2/library/re.html
```

Prepare Notebook

In [9]:
```
# import packages
from gensim import corpora
import pandas as pd
import logging
import nltk
from nltk.stem.wordnet import WordNetLemmatizer
from gensim.models import Phrases
import io
import pickle
```

In [10]:
```
# download stopwords and lemmatizer from nltk package
nltk.download("stopwords")
nltk.download("wordnet")
```

```
[nltk_data] Downloading package stopwords to C:\Users\Sebastian
[nltk_data]     Birk\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\Sebastian
[nltk_data]     Birk\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[10]:
```
True
```

In [11]:
```
# log events
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

Load and Inspect Dataset

In [12]:
```
# read data with timestamp as index
tweets = pd.read_csv("tweets.csv", encoding="latin1", parse_dates=True,
                     index_col="created", usecols=range(1,28))
```

In [13]:
```
# inspect dataframe
tweets.head()
```

Out[13]:

created	text	favoriteCount	replyToSN	truncated	replyToSID	replyToUID	statusSource	retweetCount	longitude	latitude	...	ret
2017-11-28 22:44:07	I'm at El Raval in Barcelona https://t.co/bSGA...	0	NaN	False	NaN	NaN	<a href="http://foursquare.com" rel="nofollow"...	0	2.168964	41.380936	...	
2017-11-22 19:48:53	<ed><U+00A0><U+00BC><ed><U+00B6><U+0098> @ O't...	0	NaN	False	NaN	NaN	<a href="http://instagram.com" rel="nofollow">...	0	2.168180	41.381031	...	
2017-11-21 21:58:48	Aquesta setmana publiquem una nova escapada al...	1	NaN	False	NaN	NaN	<a href="http://instagram.com" rel="nofollow">...	1	2.168721	41.380217	...	
2017-11-20 11:16:10	I'm at El Raval in Barcelona https://t.co/xz2A...	0	NaN	False	NaN	NaN	<a href="http://foursquare.com" rel="nofollow"...	0	2.168964	41.380936	...	
2017-11-20 10:08:51	Hablan catalán y es importante destacar que el...	0	NaN	False	NaN	NaN	<a href="http://instagram.com" rel="nofollow">...	0	2.168180	41.381031	...	

5 rows × 26 columns

In [14]:
```
# display dataframe info
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 23778 entries, 2017-11-28 22:44:07 to 2017-06-11 15:55:42
Data columns (total 26 columns):
text                    23778 non-null object
favoriteCount           23778 non-null int64
replyToSN               821 non-null object
truncated               23778 non-null bool
replyToSID              574 non-null float64
replyToUID              821 non-null float64
statusSource            23778 non-null object
retweetCount            23778 non-null int64
longitude               23778 non-null float64
latitude                23778 non-null float64
id_seccion              23778 non-null int64
horaPeticion            23778 non-null object
id_distrito             23778 non-null int64
grupoHora               23778 non-null object
id_seccion_xy           23778 non-null int64
favoriteCountOutlier    23778 non-null int64
retweetCountOutlier     23778 non-null int64
tweetcount              23778 non-null int64
movement                23778 non-null float64
language3               23778 non-null object
dayofweek               23778 non-null object
weeknumber              23778 non-null int64
month                   23778 non-null object
idBarrio_xy             23778 non-null int64
idBarrio                23778 non-null int64
user                    23778 non-null object
dtypes: bool(1), float64(5), int64(11), object(9)
memory usage: 4.7+ MB
```

In [15]:
```
# describe dataframe
tweets.describe()
```

Out[15]:

	favoriteCount	replyToSID	replyToUID	retweetCount	longitude	latitude	id_seccion	id_distrito	id_seccion_xy	favoriteCountOutlier
count	23778.000000	5.740000e+02	8.210000e+02	23778.000000	23778.000000	23778.000000	2.377800e+04	23778.000000	2.377800e+04	23778.000000
mean	1.200774	9.015351e+17	4.078693e+16	0.285222	2.171064	41.395325	8.019040e+08	801903.985953	7.999476e+08	0.000673
std	20.403217	5.293076e+16	1.808261e+17	4.610914	0.021190	0.014936	3.028198e+03	3.023254	3.955736e+07	0.025932
min	0.000000	1.229677e+17	7.802900e+05	0.000000	2.059243	41.332580	8.019010e+08	801901.000000	0.000000e+00	0.000000
25%	0.000000	8.862349e+17	1.195479e+08	0.000000	2.159720	41.382780	8.019020e+08	801902.000000	8.019020e+08	0.000000
50%	0.000000	9.079774e+17	3.537928e+08	0.000000	2.174778	41.395250	8.019021e+08	801902.000000	8.019021e+08	0.000000
75%	1.000000	9.232741e+17	1.028215e+09	0.000000	2.176944	41.404080	8.019060e+08	801906.000000	8.019050e+08	0.000000
max	2449.000000	9.354775e+17	9.290872e+17	567.000000	2.226620	41.465590	8.019102e+08	801910.000000	8.019102e+08	1.000000

In [16]:
```
# divide dataset according to language: extract english language
english_tweets = tweets[tweets["language3"] == "ENGLISH"].copy()
```

Data Cleaning and Preparation

Preprocess Twitter Text

In [17]:
```
# display text
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', -1)

english_tweets["text"]
```

Out[17]:
```
created
2017-11-18 19:12:28    Brew Pub to try a few of the 30 beers on offer <ed><U+00A0><U+00BD><ed><U+00B8><U+0080> #olgodbarcelo
na #ølgod #triathlontraining
 https://t.co/X3TK9OnJjs


2017-11-27 10:46:20    "Art is coming face to face with yourself" <ed><U+00A0><U+00BD><ed><U+00B9><U+0083><ed><U+00A0><U+00B
D><ed><U+00B9><U+0082> @ Parc de les Tres Xemeneies https://t.co/FedFTbDc6E


2017-11-25 20:44:52    Today's quickie @ Parc de les Tres Xemeneies https://t.co/ASKSgYfks8



2017-11-23 10:47:18    All hail the freshest seafood in town. Let's feast! <ed><U+00A0><U+00BD><ed><U+00B8><U+008B> #hadthem
ostmusselsinmylife #tripoftheyear
 https://t.co/k9civVLFzD
```

In [18]:
```
# remove links
english_tweets["text_clean"] = english_tweets["text"].str.replace(r"http\S+", "")
# remove emoticons
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"<.*>", "")
# remove punctuation, special characters etc.
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"&amp", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\.", " ")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\,", " ")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\;", " ")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\-", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\"", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace("\\\\", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace("\/", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace("\*", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"@", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\n", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\|", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"W//", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"!", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"~", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r")", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"(", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"?", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r":", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\{", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"\}", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"_", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"h:[0-9]+m:[0-9]+s", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[0-9]+", " ")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"w/", " ")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x97]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x96]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x95]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x94]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x93]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x92]+", "'")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x91]+", "")
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r"[\x85]+", "")

# reduce white spaces to 1
english_tweets["text_clean"] = english_tweets["text_clean"].str.replace(r" +", " ")
```

In [19]:
```
# display cleaned text
english_tweets["text_clean"]
```

Out[19]:
```
created
2017-11-18 19:12:28    Brew Pub to try a few of the beers on offer #olgodbarcelona #ølgod #triathlontraining

2017-11-27 10:46:20    Art is coming face to face with yourself Parc de les Tres Xemeneies

2017-11-25 20:44:52    Today's quickie Parc de les Tres Xemeneies

2017-11-23 10:47:18    All hail the freshest seafood in town Let's feast #hadthemostmusselsinmylife #tripoftheyear

2017-11-19 22:30:26    Dance to the beat of your own drum #urban #music #live #stage #graffiti #streetart

2017-11-19 10:37:25    I just finished walking km in h m s with #Endomondo #endorphins

2017-11-26 11:24:48    #blackfriday shopping #friends Then #bday party for Mallan great food margaritas and more

2017-11-25 18:52:42    Another why not La Pastisseria Barcelona

2017-11-19 12:01:56    Yesterday I got a new hairstyle With the best hair artist in the city I cut by Mychell
```

In [20]:
```
# reorder columns
english_tweets.columns
cols = ['text', 'text_clean', 'favoriteCount', 'replyToSN', 'truncated', 'replyToSID',
       'replyToUID', 'statusSource', 'retweetCount', 'longitude', 'latitude',
       'id_seccion', 'horaPeticion', 'id_distrito', 'grupoHora',
       'id_seccion_xy', 'favoriteCountOutlier', 'retweetCountOutlier',
       'tweetcount', 'movement', 'language3', 'dayofweek', 'weeknumber',
       'month', 'idBarrio_xy', 'idBarrio', 'user']

english_tweets = english_tweets[cols]
```

In order to train the topic models, 3 different pooling methods for the creation of documents are used: No Pooling (1), User Pooling (2) and Hashtag Pooling (3).

Training Documents Option 1 (No Pooling)

In [21]:
```
# treat every tweet as a different document (no pooling)
documents = english_tweets["text_clean"].tolist()
```

Training Documents Option 2 (User Pooling)

In [22]:
```
# treat all tweets by one user as one single document (user pooling)
user_combined = english_tweets[["text_clean","user"]].groupby("user")["text_clean"].apply(lambda x: "".join(x))
documents_user_pooling = user_combined.tolist()
```

Training Documents Option 3 (Hashtag Pooling)

In [23]:
```
# treat all tweets with the same hashtag as one single document (hashtag pooling)

# find all hashtags
english_tweets["hashtags"] = english_tweets["text_clean"].str.findall(r'#.*?(?=\s|$)')

# separate hashtags in columns
hashtags_tweets = pd.DataFrame(english_tweets["hashtags"].tolist(),
                               columns=["hashtag1", "hashtag2", "hashtag3", "hashtag4",
                                        "hashtag5", "hashtag6", "hashtag7", "hashtag8",
                                        "hashtag9", "hashtag10", "hashtag11", "hashtag12",
                                        "hashtag13"])

# join hashtags with tweet text
hashtags_tweets.index = english_tweets.index
hashtags_tweets = english_tweets.join(hashtags_tweets)
```

In [24]:
```
# create one dataframe with text for each hashtag column and save them in a dictionary
dict = {}
for index, item in enumerate(["hash1", "hash2", "hash3", "hash4", "hash5",
                              "hash6", "hash7", "hash8", "hash9", "hash10",
                              "hash11", "hash12", "hash13"]):
    dict[item] = hashtags_tweets[["hashtag" + str(index + 1), "text_clean"]].copy()
    dict[item].columns = ["hashtag", "text"]
    dict[item].dropna(inplace=True)

# concatenate all dataframes to one dataframe (the result is a dataframe
# where there is text for each hashtag found)
hashtags = pd.DataFrame()
for item in dict:
    hashtags = pd.concat([hashtags, dict[item]])

# combine text for each hashtag
hashtags_combined = hashtags.groupby("hashtag")["text"].apply(lambda x: "".join(x))
```

In [25]:
```
# remove some generic hashtags that cover a lot of different topics
hashtags_combined.drop(["#Barcelona", "#Catalunya", "#Spain", "#BCN", "#BARCELONA",
                        "#Espana", "#BarcelonaSpain"], inplace=True)
```

In [26]:
```
# create documents
documents_hashtag_pooling = hashtags_combined.tolist()
```

Analyze Documents

In [ ]:
```
# calculate number of terms per document
total = []
for item in documents:
    total.append(len(item.split()))
```

In [ ]:
```
# calculate total number of terms
sum = 0
for number in total:
    sum += number
```

In [ ]:
```
# calculate average number of terms per document
sum/len(documents)
```

Prepare Test Documents

The trained topic models will then be used to determine the topics of test documents.

The first objective of the research is to analyze the distribution of topics over the districts. For this purpose, district pooling is used to create the documents that will be tested.

In [27]:
```
# merge all tweets from each district (district pooling) and treat them as one single document respectively
district_combined = english_tweets[["text_clean","id_distrito"]].groupby("id_distrito")["text_clean"].apply(lambda x: "".join(x))
documents_district_pooling = district_combined.tolist()
```

In [21]:
```
# check documents
district_combined
```

Out[21]:
```
id_distrito
801901    #cure Betty Ford's in Barcelona Sunday funday wishing a could skate like a pro #iceskating #ice #winter #fun Winte
r Where en Black and White Christine The Benefit Concert Nov was a wrap Dreams come true if you work hard for it Mine came
Insider Tips on Tickets Bar by foodieling in Travel with Ling #TicketsBar #Barcelona #brodasbros #artistas #top #recomano an
ar #cracks #hiphop #tribute #jamesbrown #teatrecondal Plaça Espanya in #Barcelona #Spain #Europe #art #travel #instatravel #
instatrip #tour What a night dondiablo #DonDiablo #fcbarcelona #opium #barcelona #spain #hexagon Los Martes Now #languageexc
hange #intercambiolinguistico #funeventsbcn #meetingpeople preview #kettal #luxury #outdoor #lifestyle #barcelona #made #din
ing #chair #sofa #chaise Croissant crisis in France #watercolor #winsorandnewton Art i Col lec Claudia Vives Fierro  La chec
k list para gestionar #SocialMedia ¿añadiríais alguna tarea #RedesSociales Spanish trails #photography #barcelona #nightvisi
on #night #trails #street Thirty six taps and nothing wrong #abirradero Abirradero El cigarrillo de la tarde fumar es malo l
o sé en Gothic Quarter Barcelona In remembrance of Spain in Barcelona #CatalanIdenpendence Gothic Quarter Barcelona La esper
a desespera #sagradafamilia #cafe #history en Basílica de la Sagrada Família Ja a dins del Parlament en Parliament of Catalo
nia Buey gruposagardi en Sagardi BCN Gotic cc bync Freedom of Catalonia#LaPedrera #CasaMilà #FreedomOfCatalonia #Freedom #Ca
talonia  Insider Tips on Dirty South BCN by barcelonafood in Barcelona Food Experience Follow the white rabbit #graffiti #st
reetart #rabbit #building #city #barcelona We are too excited to be exploring a new city tomorrow Who doesn't love a long we
ekend in Tonight piece by piece as we ate it Gothic Quarter Barcelona Inlove with this narrow centuriesold stony street the
gothic church intricately designed at That's an awfully prickly prick LoveStopOficial Gothic Quarter Barcelona Thunder and l
ightning rainstorm in #Barcelona #Weather #AtLeastItsWarm #SprintToTheBar Gothic Five minutes turn the volume high #JordisLl
ibertat #Catalonia #Catalunya #Catalogne FinancialTimes BBCBreaking Jamón jeaven and other delicious Spanish goodies #Barcel
```

In [31]:
```
# check if order is right
district_combined.index # correct
```

Out[31]:
```
Int64Index([801901, 801902, 801903, 801904, 801905, 801906, 801907, 801908, 801909, 801910], dtype='int64', name='id_distrito')
```

The second objective is to look at the dynamic topic development over time. For this purpose, the dataset is divided according to time and documents are created on this basis.

Divide dataframe according to month

In [32]:
```
# sort index
sorted_tweets = english_tweets.sort_index()
```

In [33]:
```
# check first and last date
print(sorted_tweets.index[0]) # June 2017
print(sorted_tweets.index[-1]) # December 2017 (very incomplete)
```

```
2017-06-11 13:46:35
2017-12-04 21:20:25
```

In [34]:
```
# create column that contains the month of the tweets
sorted_tweets['month'] = sorted_tweets.index.month
```

In [37]:
```
# split dataframe according to month
june = sorted_tweets.loc['2017-06-01':'2017-06-30']
july = sorted_tweets.loc['2017-07-01':'2017-07-31']
august = sorted_tweets.loc['2017-08-01':'2017-08-31']
september = sorted_tweets.loc['2017-09-01':'2017-09-30']
october = sorted_tweets.loc['2017-10-01':'2017-10-31']
november = sorted_tweets.loc['2017-11-01':'2017-11-30']
december = sorted_tweets.loc['2017-12-01':'2017-12-31']
```

In [38]:
```
# count number of tweets per month
len(june)
```

Out[38]:
```
1270
```

In [45]:
```
# merge all tweets from each month and treat them as one document respectively
months_combined = english_tweets[["text_clean","month"]].groupby("month")["text_clean"].apply(lambda x: "".join(x))
documents_month_pooling = months_combined.tolist()
```

In [43]:
```
# check how many documents
len(documents_month_pooling) # should be 7
```

Out[43]:
```
7
```

In [79]:
```
# check order of documents
months_combined.index # Aug, Dec, Jul, Jun, Nov, Oct, Sep
```

Out[79]:
```
Index(['August', 'December', 'July', 'June', 'November', 'October', 'September'], dtype='object', name='month')
```

In [95]:
```
# count number of characters
len(documents_month_pooling[6])
```

Out[95]:
```
102928
```

In [85]:
```
# convert strings to wordlist
aug_words = documents_month_pooling[0].split()
dec_words = documents_month_pooling[1].split()
jul_words = documents_month_pooling[2].split()
jun_words = documents_month_pooling[3].split()
nov_words = documents_month_pooling[4].split()
oct_words = documents_month_pooling[5].split()
sept_words = documents_month_pooling[6].split()
```

In [88]:
```
# count number of words per month
print(len(jun_words))
print(len(jul_words))
print(len(aug_words))
print(len(sept_words))
print(len(oct_words))
print(len(nov_words))
print(len(dec_words))
```

```
13426
18927
12578
15347
12436
8855
380
```

In [101]:
```
# count unique words
print(len(set(jun_words)))
print(len(set(jul_words)))
print(len(set(aug_words)))
print(len(set(sept_words)))
print(len(set(oct_words)))
print(len(set(nov_words)))
print(len(set(dec_words)))
```

```
4691
6007
4416
5083
4707
3783
248
```

In [74]:
```
# merge all tweets from one district and one month and treat them as one document respectively
districts_per_month_combined = sorted_tweets[["text_clean","month","id_distrito"]].groupby(["month","id_distrito"])["text_clean"]
documents_district_per_month_pooling = districts_per_month_combined.tolist()
```

In [81]:
```
# check order of documents
districts_per_month_combined.index
```

Out[81]:
```
MultiIndex(levels=[[6, 7, 8, 9, 10, 11, 12], [801901, 801902, 801903, 801904, 801905, 801906, 801907, 801908, 801909, 801910]],
           labels=[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5,
6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 6, 7]],
           names=['month', 'id_distrito'])
```

In [50]:
```
# check how many documents
len(documents_district_per_month_pooling)
```

Out[50]: 62

In [30]:
```
# inspect dataframe
districts_per_month_combined
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

Preprocess Documents for NMF Topic Modeling Method

In [51]:
```
# create copy of no pooling documents
nmf_documents = list(documents)
```

In [52]:
```
# transform to lower case
for doc_idx, doc in enumerate(nmf_documents):
    nmf_documents[doc_idx] = nmf_documents[doc_idx].lower()
```

In [53]:
```
# delete stopwords
for doc_idx, doc in enumerate(nmf_documents):
    nmf_documents[doc_idx] = doc.replace(" year ", " ").replace(" to ", " ").replace(" on ", " ").replace(" wa ", " ").replace("
```

In [54]:
```
# display preprocessed documents
nmf_documents
```

Out[54]:
```
['brew pub try few beers offer #olgodbarcelona #ølgod #triathlontraining ',
 'art is coming face face with yourself parc les tres xemeneies ',
 "today's quickie parc les tres xemeneies ",
 "all hail freshest seafood town let's feast #hadthemostmusselsinmylife #tripoftheyear ",
 'dance beat your own drum #urban #music #live #stage #graffiti #streetart ',
 'i finished walking km h m s with #endomondo #endorphins ',
 '#blackfriday shopping #friends then #bday party mallan great food margaritas more ',
 'another why not pastisseria ',
 'yesterday i got new hairstyle with best hair artist city i cut mychell ',
 'good night tienes don jesusisnard #ootd margot house ',
 'stay warm pic x jesusisnard #sunday margot house ',
 'there is nothing better than waking up this saturdays margothousebarcelona wearing ',
 'drinking pumpkin catalanbrewery cocovailbh ',
 'drinking leyenda cervezadougalls cocovailbh ',
 'barcelonas night clinicasden #event #night #night #clinicasden #sonrisas ',
 'brama crosmaslounge w terraferma ',
 'trying new jobs perspectives #telenoticies #tv #tvnews #tn #tvset #television ',
 'from minute we emet bri nick we were bowled over how friendly easygoing ',
 "#saturday it's time #ride #bike bhbikes dhbsport cold weather let's face ",
```

Save Training and Test Documents

In [55]:
```
with io.open('documents.txt', 'w', encoding='utf-8') as f:
    for item in documents:
        f.write(item + "\n")
with io.open('documents_user_pooling.txt', 'w', encoding='utf-8') as f:
    for item in documents_user_pooling:
        f.write(item + "\n")
with io.open('documents_hashtag_pooling.txt', 'w', encoding='utf-8') as f:
    for item in documents_hashtag_pooling:
        f.write(item + "\n")
with io.open('documents_district_pooling.txt', 'w', encoding='utf-8') as f:
    for item in documents_district_pooling:
        f.write(item + "\n")
with io.open('documents_month_pooling.txt', 'w', encoding='utf-8') as f:
    for item in documents_month_pooling:
        f.write(item + "\n")
with io.open('documents_district_per_month_pooling.txt', 'w', encoding='utf-8') as f:
    for item in documents_district_per_month_pooling:
        f.write(item + "\n")

with open('documents_no_pooling.p', 'wb') as fp:
    pickle.dump(documents, fp)

with open('documents_user_pooling.p', 'wb') as fp:
    pickle.dump(documents_user_pooling, fp)

with open('documents_hashtag_pooling.p', 'wb') as fp:
    pickle.dump(documents_hashtag_pooling, fp)

with open('nmf_documents_no_pooling.p', 'wb') as fp:
    pickle.dump(nmf_documents, fp)
```

In [56]:
```
nmf_documents
```

Out[56]:
```
['brew pub try few beers offer #olgodbarcelona #ølgod #triathlontraining ',
 'art is coming face face with yourself parc les tres xemeneies ',
 "today's quickie parc les tres xemeneies ",
 "all hail freshest seafood town let's feast #hadthemostmusselsinmylife #tripoftheyear ",
 'dance beat your own drum #urban #music #live #stage #graffiti #streetart ',
 'i finished walking km h m s with #endomondo #endorphins ',
 '#blackfriday shopping #friends then #bday party mallan great food margaritas more ',
 'another why not pastisseria ',
 'yesterday i got new hairstyle with best hair artist city i cut mychell ',
 'good night tienes don jesusisnard #ootd margot house ',
 'stay warm pic x jesusisnard #sunday margot house ',
 'there is nothing better than waking up this saturdays margothousebarcelona wearing ',
 'drinking pumpkin catalanbrewery cocovailbh ',
 'drinking leyenda cervezadougalls cocovailbh ',
 'barcelonas night clinicasden #event #night #night #clinicasden #sonrisas ',
 'brama crosmaslounge w terraferma ',
 'trying new jobs perspectives #telenoticies #tv #tvnews #tn #tvset #television ',
 'from minute we emet bri nick we were bowled over how friendly easygoing ',
 "#saturday it's time #ride #bike bhbikes dhbsport cold weather let's face ",
```

Tokenize Training Documents

In [57]:
```
# we can simply tokenize by space thanks to the previous preprocessing
texts_no_pooling = [[word for word in document.lower().split()]
          for document in documents]

texts_user_pooling = [[word for word in document.lower().split()]
          for document in documents_user_pooling]

texts_hashtag_pooling = [[word for word in document.lower().split()]
          for document in documents_hashtag_pooling]
```

Save Unpreprocessed Tokenized Training Documents

In [58]:
```
with open('tokenized_documents_no_pooling_unpp.p', 'wb') as fp:
    pickle.dump(texts_no_pooling, fp)

with open('tokenized_documents_user_pooling_unpp.p', 'wb') as fp:
    pickle.dump(texts_user_pooling, fp)

with open('tokenized_documents_hashtag_pooling_unpp.p', 'wb') as fp:
    pickle.dump(texts_hashtag_pooling, fp)
```

Further Preprocessing of Training Documents after Tokenization

In [59]:
```
# remove numbers, but not words that contain numbers.
texts_no_pooling = [[token for token in doc if not token.isnumeric()] for doc in texts_no_pooling]
texts_user_pooling = [[token for token in doc if not token.isnumeric()] for doc in texts_user_pooling]
texts_hashtag_pooling = [[token for token in doc if not token.isnumeric()] for doc in texts_hashtag_pooling]
```

In [60]:
```
# remove words that are only one character.
texts_no_pooling = [[token for token in doc if len(token) > 1] for doc in texts_no_pooling]
texts_user_pooling = [[token for token in doc if len(token) > 1] for doc in texts_user_pooling]
texts_hashtag_pooling = [[token for token in doc if len(token) > 1] for doc in texts_hashtag_pooling]
```

In [61]:
```
# lemmatize all words in all documents.
lemmatizer = WordNetLemmatizer()
texts_no_pooling = [[lemmatizer.lemmatize(token) for token in doc] for doc in texts_no_pooling]
texts_user_pooling = [[lemmatizer.lemmatize(token) for token in doc] for doc in texts_user_pooling]
texts_hashtag_pooling = [[lemmatizer.lemmatize(token) for token in doc] for doc in texts_hashtag_pooling]
```

In [62]:
```
# # ignore this part! computing bigrams did not improve models but made them worse!

# # compute bigrams
# # add bigrams and trigrams to docs (only ones that appear 5 times or more)
# bigram = Phrases(texts_no_pooling, min_count=10)
# for idx in range(len(texts_no_pooling)):
#     for token in bigram[texts_no_pooling[idx]]:
#         if '_' in token:
#             # Token is a bigram, add to document.
#             texts_no_pooling[idx].append(token)
#
# bigram = Phrases(texts_user_pooling, min_count=10)
# for idx in range(len(texts_user_pooling)):
#     for token in bigram[texts_user_pooling[idx]]:
#         if '_' in token:
#             # Token is a bigram, add to document.
#             texts_user_pooling[idx].append(token)
#
# bigram = Phrases(texts_hashtag_pooling, min_count=10)
# for idx in range(len(texts_hashtag_pooling)):
#     for token in bigram[texts_hashtag_pooling[idx]]:
#         if '_' in token:
#             # Token is a bigram, add to document.
#             texts_hashtag_pooling[idx].append(token)
```

Save Tokenized Training Documents

In [63]:
```
with open('tokenized_documents_no_pooling.p', 'wb') as fp:
    pickle.dump(texts_no_pooling, fp)

with open('tokenized_documents_user_pooling.p', 'wb') as fp:
    pickle.dump(texts_user_pooling, fp)

with open('tokenized_documents_hashtag_pooling.p', 'wb') as fp:
    pickle.dump(texts_hashtag_pooling, fp)
```

Refine and Vectorize Corpora

In [64]:
```
# define function to refine and vectorize corpus
# (remove stopwords, very frequent and very infrequent words etc.)

# define stopwords
stpwords = "for a of the and to in at by one #yo #el day get españa #yourup #españa #repost yo el since still never thank two thi

def nltk_stopwords():
    return set(nltk.corpus.stopwords.words('english'))

def prep_corpus(docs,
                additional_stopwords=set(stpwords),
                no_below=2, no_above=0.5,
                dictionary_name='tourism.dict', corpus_name='tourism.mm'):
    print('Building dictionary...')
    dictionary = corpora.Dictionary(docs)
    stopwords = nltk_stopwords().union(additional_stopwords)
    stopword_ids = map(dictionary.token2id.get, stopwords)
    dictionary.filter_tokens(stopword_ids)
    dictionary.compactify()
    dictionary.filter_extremes(no_below=no_below, no_above=no_above, keep_n=None)
    dictionary.compactify()
    dictionary.save(dictionary_name)  # store the dictionary, for future reference

    print('Building corpus...')
    corpus = [dictionary.doc2bow(doc) for doc in docs]
    corpora.MmCorpus.serialize(corpus_name, corpus)  # store to disk, for later use

    return (corpus, dictionary)
```

In [65]:
```
# run function to vectorize corpora
corpus_no_pooling = prep_corpus(texts_no_pooling,
                                dictionary_name="tourism_no_pooling.dict",
                                corpus_name="tourism_no_pooling.mm")[0]
dictionary_no_pooling = prep_corpus(texts_no_pooling,
                                    dictionary_name="tourism_no_pooling.dict",
                                    corpus_name="tourism_no_pooling.mm")[1]

corpus_user_pooling = prep_corpus(texts_user_pooling,
                                    dictionary_name="tourism_user_pooling.dict",
                                    corpus_name="tourism_user_pooling.mm")[0]
dictionary_user_pooling = prep_corpus(texts_user_pooling,
                                    dictionary_name="tourism_user_pooling.dict",
                                    corpus_name="tourism_user_pooling.mm")[1]

corpus_hashtag_pooling = prep_corpus(texts_hashtag_pooling,
                                    dictionary_name="tourism_hashtag_pooling.dict",
                                    corpus_name="tourism_hashtag_pooling.mm")[0]
dictionary_hashtag_pooling = prep_corpus(texts_hashtag_pooling,
                                    dictionary_name="tourism_hashtag_pooling.dict",
                                    corpus_name="tourism_hashtag_pooling.mm")[1]
```

```
2018-10-05 07:05:26,049 : INFO : adding document #0 to Dictionary(0 unique tokens: [])
Building dictionary...
2018-10-05 07:05:26,264 : INFO : built Dictionary(15443 unique tokens: ['#olgodbarcelona', '#triathlontraining', '#ølgod',
'beer', 'brew']...) from 7633 documents (total 78801 corpus positions)
2018-10-05 07:05:26,323 : INFO : discarding 10451 tokens: [('#olgodbarcelona', 1), ('#triathlontraining', 1), ('quickie',
1), ('#hadthemostmusselsinmylife', 1), ('#tripoftheyear', 1), ('freshest', 1), ('hail', 1), ('mallan', 1), ('mychell', 1),
('tienes', 1)]...
2018-10-05 07:05:26,325 : INFO : keeping 4790 tokens which were in no less than 2 and no more than 3816 (=50.0%) documents
2018-10-05 07:05:26,342 : INFO : resulting dictionary: Dictionary(4790 unique tokens: ['#ølgod', 'beer', 'brew', 'offer', 'p
ub']...)
2018-10-05 07:05:26,353 : INFO : saving Dictionary object under tourism_no_pooling.dict, separately None
2018-10-05 07:05:26,360 : INFO : saved tourism_no_pooling.dict
2018-10-05 07:05:26,471 : INFO : storing corpus in Matrix Market format to tourism_no_pooling.mm
2018-10-05 07:05:26,475 : INFO : saving sparse matrix to tourism_no_pooling.mm
2018-10-05 07:05:26,478 : INFO : PROGRESS: saving document #0
2018-10-05 07:05:26,517 : INFO : PROGRESS: saving document #1000
2018-10-05 07:05:26,537 : INFO : PROGRESS: saving document #2000
2018-10-05 07:05:26,569 : INFO : PROGRESS: saving document #3000
```

Apply Function to Preprocess Test Documents (Before Testing Them with Topic Models)

This function has to include all the same steps that were applied to the training documents!

In [66]:
```
# define function
def preprocess(docs):
    ''' Conduct all preprocessing steps that are conducted to train the LDA model'''

    # tokenize documents
    tokenized = [[word for word in document.lower().split()]
          for document in docs]

    # remove words that are only one character
    tokenized = [[token for token in doc if len(token) > 1] for doc in tokenized]

    # lemmatize all words
    lemmatizer = WordNetLemmatizer()
    lemmatized = [[lemmatizer.lemmatize(token) for token in doc] for doc in tokenized]

    # define stopwords
    stpwords = "for a of the and to in at by one #yo #el day get #yourup españa #españa #repost yo el since still never thank two

    # get stopwords from nltk
    def nltk_stopwords():
        return set(nltk.corpus.stopwords.words('english'))

    # combine stopwords
    stopwords = nltk_stopwords().union(stpwords)

    # remove stopwords
    preprocessed = [[token for token in document if token not in stopwords] for document in lemmatized]

    return preprocessed
```

In [67]:
```
# apply function to test documents
texts_district_pooling = preprocess(documents_district_pooling)
texts_month_pooling = preprocess(documents_month_pooling)
texts_district_per_month_pooling = preprocess(documents_district_per_month_pooling)
```

Save Preprocessed Test Documents

In [68]:
```
with open('tokenized_documents_district_pooling.p', 'wb') as fp:
    pickle.dump(texts_district_pooling, fp)

with open('tokenized_documents_month_pooling.p', 'wb') as fp:
    pickle.dump(texts_month_pooling, fp)

with open('tokenized_documents_district_per_month_pooling.p', 'wb') as fp:
    pickle.dump(texts_district_per_month_pooling, fp)
```

In [69]:
```
# # ignore this part! just example code!

# # map tokens to ids
# print(dictionary_no_pooling.token2id)
# print(dictionary_user_pooling.token2id)
# print(dictionary_hashtag_pooling.token2id)
```

In [70]:
```
# # ignore this part! just example code!

# # convert new document to vector
# new_doc = "Sagrada Familia is amazing"
# new_vec_no_pooling = dictionary_no_pooling.doc2bow(new_doc.lower().split())
# print(new_vec_no_pooling)
```

In [71]:
```
# # ignore this part! not needed for dataset!

# # corpus streaming: one document at a time
# class MyCorpus(object):
#     def __iter__(self):
#         for line in open("corpus_no_pooling.txt"):
#             # assume there's one document per line, tokens separated by whitespace
#             yield dictionary.doc2bow(line.lower().split())
#
# corpus_memory_friendly = MyCorpus()  # doesn't load the corpus into memory!
# print(corpus_memory_friendly)
#
# for vector in corpus_memory_friendly:  # load one vector into memory at a time
#     print(vector)
```

Jupyter Notebook 2: Train LDA models

References

In [3]:
```
# https://radimrehurek.com/gensim/tut2.html#Gensim
# https://markroxor.github.io/gensim/static/notebooks/lda_training_tips.html
# https://datascienceplus.com/evaluation-of-topic-modeling-topic-coherence/
# https://pypi.org/project/pyLDAvis/1.0.0/
# https://gist.github.com/tokestermw/3588e6fbbb2f03f89798
# https://stackoverflow.com/questions/11162402/lda-topic-modeling-training-and-testing
# https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
```

Prepare Notebook

In [4]:
```
# import packages
import os.path
from gensim import corpora, models
import logging
import pickle
from gensim.models import CoherenceModel
import matplotlib.pyplot as plt
import pyLDAvis.gensim
import pyLDAvis
from wordcloud import WordCloud
```

```
C:\Users\Sebastian Birk\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [5]:
```
# display plots within notebook
%matplotlib inline
```

In [6]:
```
# Log events
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

Load Vectorized Corpora and Dictionaries

In [7]:
```
# load no pooling corpus
if (os.path.exists("tourism_no_pooling.dict")):
    dictionary_no_pooling = corpora.Dictionary.load('tourism_no_pooling.dict')
    corpus_no_pooling = corpora.MmCorpus('tourism_no_pooling.mm')
    print("Vectorized no pooling corpus loaded!")
else:
    print("Please run preprocessing script first!")

# load user pooling corpus
if (os.path.exists("tourism_user_pooling.dict")):
    dictionary_user_pooling = corpora.Dictionary.load('tourism_user_pooling.dict')
    corpus_user_pooling = corpora.MmCorpus('tourism_user_pooling.mm')
    print("Vectorized user pooling corpus loaded!")
else:
    print("Please run preprocessing script first!")

# load hashtag pooling corpus
if (os.path.exists("tourism_hashtag_pooling.dict")):
    dictionary_hashtag_pooling = corpora.Dictionary.load('tourism_hashtag_pooling.dict')
    corpus_hashtag_pooling = corpora.MmCorpus('tourism_hashtag_pooling.mm')
    print("Vectorized hashtag pooling corpus loaded!")
else:
    print("Please run preprocessing script first!")
```

```
2018-10-03 07:51:28,936 : INFO : loading Dictionary object from tourism_no_pooling.dict
2018-10-03 07:51:28,944 : INFO : loaded tourism_no_pooling.dict
2018-10-03 07:51:28,951 : INFO : loaded corpus index from tourism_no_pooling.mm.index
2018-10-03 07:51:28,953 : INFO : initializing cython corpus reader from tourism_no_pooling.mm
2018-10-03 07:51:28,956 : INFO : accepted corpus with 7633 documents, 4790 features, 37016 non-zero entries
2018-10-03 07:51:28,959 : INFO : loading Dictionary object from tourism_user_pooling.dict
2018-10-03 07:51:28,966 : INFO : loaded tourism_user_pooling.dict
2018-10-03 07:51:28,971 : INFO : loaded corpus index from tourism_user_pooling.mm.index
2018-10-03 07:51:28,974 : INFO : initializing cython corpus reader from tourism_user_pooling.mm
2018-10-03 07:51:28,978 : INFO : accepted corpus with 4424 documents, 3990 features, 30119 non-zero entries
2018-10-03 07:51:28,982 : INFO : loading Dictionary object from tourism_hashtag_pooling.dict
2018-10-03 07:51:28,990 : INFO : loaded tourism_hashtag_pooling.dict
2018-10-03 07:51:28,994 : INFO : loaded corpus index from tourism_hashtag_pooling.mm.index
2018-10-03 07:51:28,996 : INFO : initializing cython corpus reader from tourism_hashtag_pooling.mm
2018-10-03 07:51:28,999 : INFO : accepted corpus with 6040 documents, 9330 features, 78970 non-zero entries
```

```
Vectorized no pooling corpus loaded!
Vectorized user pooling corpus loaded!
Vectorized hashtag pooling corpus loaded!
```

Load Tokenized Documents

In [8]:
```
with open ('tokenized_documents_no_pooling.p', 'rb') as fp:
    tokenized_documents_no_pooling = pickle.load(fp)
with open ('tokenized_documents_user_pooling.p', 'rb') as fp:
    tokenized_documents_user_pooling = pickle.load(fp)
with open ('tokenized_documents_hashtag_pooling.p', 'rb') as fp:
    tokenized_documents_hashtag_pooling = pickle.load(fp)
```

Implement LDA Models with Different Pooling Methods

Two evaluation metrics for topic models come to mind: coherence values and perplexity. Coherence values will be used to evaluate different LDA models (varying the number of topics) as this metric tends to favor better human interpretable topics (which is the objective of this research). The number of topics will be limited to 8 to avoid too much granularity. However, sometimes the highest coherence values do not give the most human interpretable topics. Visualization of the topic models can additionally help to understand and interprete the topics. The c_v measure will be used as a coherence measure to evaluate the LDA models.

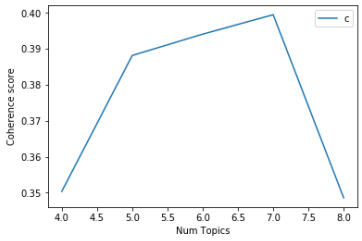
In [9]:
```
# Define function to train various LDA models with different number of topics
# and evaluate their coherence values (choose the number of topics with the highest coherence value)
def compute_coherence_values(dictionary, corpus, texts, limit=9, start=4, step=1):
    """
    Compute c_v coherence for various number of topics

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    texts : List of input texts
    limit : Max num of topics

    Returns:
    -------
    model_list : List of LDA topic models
    coherence_values : Coherence values corresponding to the LDA model with respective number of topics
    """

    coherence_values = []
    model_list = []
    model_topics = []

    for num_topics in range(start, limit, step):
        model= models.LdaModel(corpus=corpus, id2word=dictionary, alpha='auto', eta='auto',
                               eval_every=1, iterations=400, passes=20, num_topics=num_topics)
        model_list.append(model)

        model_topics = model.show_topics(formatted=False)

        model_topics = [[word for word, prob in topic] for topicid, topic in model_topics]

        coherencemodel = CoherenceModel(topics=model_topics, texts=texts, dictionary=dictionary, window_size=10)
        coherence_values.append(coherencemodel.get_coherence())

    return (model_list, coherence_values)
```

No Pooling

In [24]:
```
# train and evaluate different no pooling models by running the function
no_pooling_models = compute_coherence_values(dictionary=dictionary_no_pooling,
                            corpus=corpus_no_pooling, texts=tokenized_documents_no_pooling)
```

```
2018-10-03 08:04:29,558 : INFO : -9.682 per-word bound, 821.5 perplexity estimate based on a held-out corpus of 2000 documents with 9957 words
2018-10-03 08:04:29,560 : INFO : PROGRESS: pass 0, at document #4000/7633
2018-10-03 08:04:30,206 : INFO : optimized alpha [0.17919257, 0.2260358, 0.17883387, 0.18693891]
2018-10-03 08:04:30,207 : INFO : merging changes from 2000 documents into a model of 7633 documents
2018-10-03 08:04:30,212 : INFO : topic #0 (0.179): 0.011*"milà" + 0.011*"time" + 0.011*"pedreracasa" + 0.007*"gothic" + 0.007*"#sagradafamilia" + 0.007*"#
arter" + 0.006*"amazing" + 0.005*"#beach" + 0.005*"#gaudi"
2018-10-03 08:04:30,215 : INFO : topic #1 (0.226): 0.065*"sagrada" + 0.055*"familia" + 0.024*"#sagradafamilia" + 0.019*"gothic" + 0.018*"quarter" + 0.010*
avel" + 0.007*"#gaudi" + 0.006*"basílica" + 0.006*"#graffiti"
2018-10-03 08:04:30,217 : INFO : topic #2 (0.179): 0.018*"drinking" + 0.007*"city" + 0.006*"way" + 0.006*"place" + 0.006*"parc" + 0.006*"sagrada" + 0.006*
oday" + 0.005*"best" + 0.005*"basílica"
2018-10-03 08:04:30,220 : INFO : topic #3 (0.187): 0.044*"sagrada" + 0.027*"basílica" + 0.024*"família" + 0.015*"familia" + 0.011*"night" + 0.008*"beach"
+ 0.006*"inside" + 0.005*"#sagradafamilia" + 0.005*"amazing"
2018-10-03 08:04:30,222 : INFO : topic diff=2.260507, rho=0.707107
2018-10-03 08:04:30,997 : INFO : -9.473 per-word bound, 710.7 perplexity estimate based on a held-out corpus of 2000 documents with 9649 words
2018-10-03 08:04:30,998 : INFO : PROGRESS: pass 0, at document #6000/7633
2018-10-03 08:04:31,535 : INFO : optimized alpha [0.17872371, 0.23167226, 0.17940545, 0.18625566]
2018-10-03 08:04:31,537 : INFO : merging changes from 2000 documents into a model of 7633 documents
2018-10-03 08:04:31,543 : INFO : topic #0 (0.179): 0.013*"milà" + 0.011*"pedreracasa" + 0.010*"time" + 0.009*"#summer" + 0.007*"#beach" + 0.006*"gothic" +
0.005*"casa" + 0.005*"#gaudi" + 0.005*"#sagradafamilia"
```

In [41]:
```
# display the coherence score of the different models
model_list_no_pooling = no_pooling_models[0]
coherence_values_no_pooling = no_pooling_models[1]

limit=9; start=4; step=1;
x = range(start, limit, step)
_ = plt.plot(x, coherence_values_no_pooling)
_ = plt.xlabel("Num Topics")
_ = plt.ylabel("Coherence score")
_ = plt.legend(("coherence_values"), loc='best')
_ = plt.savefig("no_pooling_coherence_scores")
_ = plt.show()
```



Choose the model with the highest coherence score (7 topics).

In [42]:
```
# print topics of model with highest coherence score
lda_model_no_pooling = model_list_no_pooling[3] # 7 topics model
_ = lda_model_no_pooling.print_topics()
```

```
2018-10-03 08:21:20,589 : INFO : topic #0 (0.131): 0.025*"milà" + 0.022*"pedreracasa" + 0.019*"ramblas" + 0.016*"time" + 0.012*"night" + 0.012*"sant" + 0.011*"can't" + 0.008
*"see" + 0.008*"joan" + 0.008*"got"
2018-10-03 08:21:20,592 : INFO : topic #1 (0.130): 0.044*"drinking" + 0.032*"beach" + 0.023*"barceloneta" + 0.019*"playa" + 0.017*"parc" + 0.014*"platja" + 0.012*"city" + 0.
011*"#endomondo" + 0.011*"marbella" + 0.010*"km"
2018-10-03 08:21:20,594 : INFO : topic #2 (0.205): 0.134*"sagrada" + 0.086*"familia" + 0.045*"basílica" + 0.042*"#sagradafamilia" + 0.036*"família" + 0.019*"#gaudi" + 0.017
*"beautiful" + 0.010*"gaudi" + 0.008*"#architecture" + 0.008*"amazing"
2018-10-03 08:21:20,596 : INFO : topic #3 (0.123): 0.018*"hotel" + 0.012*"casa" + 0.007*"l'aquàrium" + 0.007*"night" + 0.007*"rooftop" + 0.006*"barca" + 0.006*"love" + 0.006
*"shōko" + 0.005*"meet" + 0.005*"i'll"
2018-10-03 08:21:20,599 : INFO : topic #4 (0.121): 0.022*"#streetart" + 0.015*"festival" + 0.015*"#graffiti" + 0.012*"bcnfashion" + 0.012*"#arteurbano" + 0.011*"summer" + 0.
010*"#digerible" + 0.009*"tonight" + 0.009*"fashion" + 0.009*"really"
2018-10-03 08:21:20,602 : INFO : topic #5 (0.151): 0.070*"gothic" + 0.062*"quarter" + 0.021*"#summer" + 0.017*"#love" + 0.014*"#friends" + 0.011*"#beach" + 0.008*"#sun" + 0.
007*"#art" + 0.007*"new" + 0.007*"#travel"
2018-10-03 08:21:20,604 : INFO : topic #6 (0.132): 0.014*"best" + 0.012*"place" + 0.011*"happy" + 0.009*"life" + 0.009*"tapa" + 0.009*"love" + 0.009*"bar" + 0.008*"made" +
0.008*"great" + 0.008*"ticket"
```

Visualize No Pooling Model

In [43]:
```
pyLDAvis.enable_notebook()
vis_np = pyLDAvis.gensim.prepare(lda_model_no_pooling, corpus_no_pooling, dictionary_no_pooling)
```

In [44]:
```
vis_np
```

Out[44]:

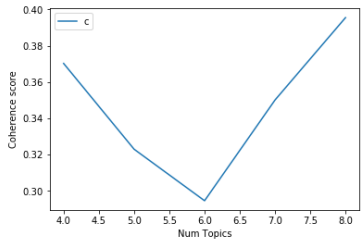
Although a topic trend is already visible in the no pooling model, the topics are a little bit mixed up and could be more interpretable. This finding can be attributed to the shortness of tweets.

User Pooling

In [31]:
```
# train and evaluate different user pooling models by running the function
user_pooling_models = compute_coherence_values(dictionary=dictionary_user_pooling,
                        corpus=corpus_user_pooling, texts=tokenized_documents_user_pooling)
```

```
2018-10-03 08:12:59,332 : INFO : using autotuned alpha, starting with [0.25, 0.25, 0.25, 0.25]
2018-10-03 08:12:59,336 : INFO : using serial LDA version on this node
2018-10-03 08:12:59,342 : INFO : running online (multi-pass) LDA training, 4 topics, 20 passes over the supplied corpus of 4424 documents, updating model
uments, evaluating perplexity every 2000 documents, iterating 400x with a convergence threshold of 0.001000
2018-10-03 08:13:01,667 : INFO : -9.113 per-word bound, 553.6 perplexity estimate based on a held-out corpus of 2000 documents with 23841 words
2018-10-03 08:13:01,668 : INFO : PROGRESS: pass 0, at document #2000/4424
2018-10-03 08:13:03,921 : INFO : optimized alpha [0.16434142, 0.15121301, 0.16912088, 0.16412929]
2018-10-03 08:13:03,922 : INFO : merging changes from 2000 documents into a model of 4424 documents
2018-10-03 08:13:03,928 : INFO : topic #0 (0.164): 0.022*"sagrada" + 0.019*"gothic" + 0.017*"basílica" + 0.016*"quarter" + 0.013*"família" + 0.009*"night"
+ 0.005*"new" + 0.005*"#travel" + 0.005*"ramblas"
2018-10-03 08:13:03,931 : INFO : topic #1 (0.151): 0.023*"gothic" + 0.022*"quarter" + 0.010*"#photodrinking" + 0.009*"beach" + 0.008*"#travel" + 0.007*"lo
a" + 0.007*"cervecita" + 0.007*"city" + 0.006*"familia"
2018-10-03 08:13:03,933 : INFO : topic #2 (0.169): 0.037*"sagrada" + 0.025*"familia" + 0.013*"drinking" + 0.011*"time" + 0.011*"#sagradafamilia" + 0.009*"
asílica" + 0.008*"#gaudi" + 0.006*"gaudi" + 0.005*"milà"
2018-10-03 08:13:03,935 : INFO : topic #3 (0.164): 0.032*"sagrada" + 0.023*"familia" + 0.015*"#sagradafamilia" + 0.010*"#streetart" + 0.008*"#graffiti" +
+ 0.007*"basílica" + 0.007*"#photography" + 0.006*"milà" + 0.005*"pedreracasa"
2018-10-03 08:13:03,937 : INFO : topic diff=2.696676, rho=1.000000
2018-10-03 08:13:04,911 : INFO : -8.662 per-word bound, 405.1 perplexity estimate based on a held-out corpus of 2000 documents with 9561 words
```

In [36]:
```
# display the coherence score of the different models
model_list_user_pooling = user_pooling_models[0]
coherence_values_user_pooling = user_pooling_models[1]

limit=9; start=4; step=1;
x = range(start, limit, step)
_ = plt.plot(x, coherence_values_user_pooling)
_ = plt.xlabel("Num Topics")
_ = plt.ylabel("Coherence score")
_ = plt.legend(("coherence_values"), loc='best')
_ = plt.savefig("user_pooling_coherence_scores")
_ = plt.show()
```



Choose the model with the highest coherence score (8 topics).

In [46]:
```
# print topics of model with highest coherence score
lda_model_user_pooling = model_list_user_pooling[4] # 8 topics model
_ = lda_model_user_pooling.print_topics()
```

```
2018-10-03 08:22:00,050 : INFO : topic #0 (0.188): 0.116*"sagrada" + 0.073*"familia" + 0.039*"basílica" + 0.035*"#sagradafamilia" + 0.032*"família" + 0.016*"#gaudi" + 0.012
*"gaudi" + 0.010*"beautiful" + 0.009*"amazing" + 0.007*"place"
2018-10-03 08:22:00,053 : INFO : topic #1 (0.083): 0.006*"gothic" + 0.077*"quarter" + 0.012*"#endomondo" + 0.011*"km" + 0.011*"#endorphins" + 0.010*"dream" + 0.009*"finishe
d" + 0.009*"time" + 0.008*"street" + 0.008*"food"
2018-10-03 08:22:00,056 : INFO : topic #2 (0.078): 0.023*"milà" + 0.020*"pedreracasa" + 0.013*"festival" + 0.013*"time" + 0.011*"sound" + 0.009*"primavera" + 0.007*"#primave
rasound" + 0.007*"#amazing" + 0.006*"cool" + 0.006*"got"
2018-10-03 08:22:00,058 : INFO : topic #3 (0.061): 0.013*"best" + 0.010*"#photography" + 0.008*"e" + 0.007*"today" + 0.007*"night" + 0.006*"new" + 0.006*"road" + 0.006*"cock
tail" + 0.006*"took" + 0.006*"catalunya"
2018-10-03 08:22:00,061 : INFO : topic #4 (0.070): 0.024*"#summer" + 0.011*"bcnfashion" + 0.010*"#sun" + 0.010*"blue" + 0.009*"good" + 0.008*"night" + 0.007*"avinguda" + 0.0
07*"opium" + 0.007*"art" + 0.007*"#friends"
2018-10-03 08:22:00,063 : INFO : topic #5 (0.074): 0.014*"drinking" + 0.013*"ramblas" + 0.013*"fashion" + 0.009*"recinte" + 0.008*"sant" + 0.008*"night" + 0.007*"modernista"
+ 0.006*"time" + 0.006*"watching" + 0.006*"pau"
2018-10-03 08:22:00,066 : INFO : topic #6 (0.081): 0.022*"parc" + 0.017*"#streetart" + 0.011*"#graffiti" + 0.011*"guell" + 0.010*"#photodrinking" + 0.010*"drinking" + 0.008
*"new" + 0.007*"laberint" + 0.007*"#arteurbano" + 0.007*"l'aquàrium"
2018-10-03 08:22:00,068 : INFO : topic #7 (0.096): 0.038*"barceloneta" + 0.036*"playa" + 0.031*"beach" + 0.016*"platja" + 0.012*"time" + 0.011*"love" + 0.011*"city" + 0.010
*"week" + 0.010*"marbella" + 0.010*"next"
```

Visualize User Pooling Model

In [47]:
```
pyLDAvis.enable_notebook()
vis_up = pyLDAvis.gensim.prepare(lda_model_user_pooling, corpus_user_pooling, dictionary_user_pooling)
```

In [48]:
```
vis_up
```

Out[48]:

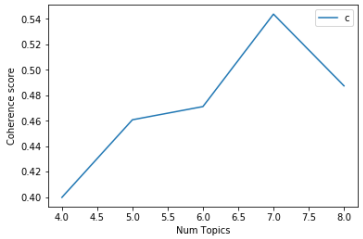
The results of the user pooling model look similar to the no pooling model. However, topics are even more mixed up and less interpretable since users tend to tweet about different topics.

Hashtag Pooling

In [10]:
```
# train and evaluate different hashtag pooling models by running the function
hashtag_pooling_models = compute_coherence_values(dictionary=dictionary_hashtag_pooling,
                         corpus=corpus_hashtag_pooling, texts=tokenized_documents_hashtag_pooling)
```

```
7*"#artecallejero" + 0.007*"#city" + 0.007*"arte" + 0.007*"urbano"
2018-10-03 07:51:49,757 : INFO : topic #1 (0.172): 0.023*"#sagradafamilia" + 0.016*"sagrada" + 0.012*"#architecture" + 0.010*"familia" + 0.010*"#art" + 0.
9*"#travel" + 0.008*"#summer" + 0.007*"gothic" + 0.006*"#love"
2018-10-03 07:51:49,760 : INFO : topic #2 (0.153): 0.008*"#catalunya" + 0.008*"#sport" + 0.008*"#women" + 0.007*"#training" + 0.007*"#lifestyle" + 0.007*
*"#spoonie" + 0.007*"#neversettle" + 0.006*"#workout" + 0.006*"#fit"
2018-10-03 07:51:49,762 : INFO : topic #3 (0.168): 0.019*"#sagradafamilia" + 0.018*"sagrada" + 0.010*"basílica" + 0.009*"#catalonia" + 0.009*"#gaudi" + 0.
07*"#travel" + 0.006*"#beach" + 0.006*"#church" + 0.005*"familia"
2018-10-03 07:51:49,764 : INFO : topic diff=2.945522, rho=1.000000
2018-10-03 07:51:50,725 : INFO : -9.825 per-word bound, 907.2 perplexity estimate based on a held-out corpus of 2000 documents with 33997 words
2018-10-03 07:51:50,726 : INFO : PROGRESS: pass 0, at document #4000/6040
2018-10-03 07:51:51,476 : INFO : optimized alpha [0.12321521, 0.15629904, 0.12627287, 0.14378324]
2018-10-03 07:51:51,479 : INFO : merging changes from 2000 documents into a model of 6040 documents
2018-10-03 07:51:51,486 : INFO : topic #0 (0.123): 0.019*"#graffiti" + 0.017*"#streetart" + 0.010*"#arteurbano" + 0.009*"#digerible" + 0.009*"#art" + 0.00
rbano" + 0.006*"#arteenlacalle" + 0.006*"#artecallejero" + 0.006*"#photooftheday"
2018-10-03 07:51:51,488 : INFO : topic #1 (0.156): 0.018*"#sagradafamilia" + 0.014*"sagrada" + 0.014*"#gaudi" + 0.013*"#love" + 0.010*"familia" + 0.009*"#
summer" + 0.009*"#travel" + 0.009*"#art" + 0.007*"#architecture"
2018-10-03 07:51:51,491 : INFO : topic #2 (0.126): 0.008*"#love" + 0.006*"#endomondo" + 0.006*"km" + 0.006*"#endorphins" + 0.006*"#friends" + 0.006*"#gato
le" + 0.005*"#catalunya" + 0.004*"#summer" + 0.004*"finished"
```

In [22]:
```
# display the coherence score of the different models
model_list_hashtag_pooling = hashtag_pooling_models[0]
coherence_values_hashtag_pooling = hashtag_pooling_models[1]

limit=9; start=4; step=1;
x = range(start, limit, step)
_ = plt.plot(x, coherence_values_hashtag_pooling)
_ = plt.xlabel("Num Topics")
_ = plt.ylabel("Coherence score")
_ = plt.legend(("coherence_values"), loc='best')
_ = plt.savefig("hashtag_pooling_coherence_scores")
_ = plt.show()
```



Choose the model with the highest coherence score (7 topics).

In [17]:
```
# print topics of model with highest coherence score
lda_model_hashtag_pooling = model_list_hashtag_pooling[3] # 7 topics
_ = lda_model_hashtag_pooling.print_topics()
```

```
2018-10-03 08:01:25,291 : INFO : topic #0 (0.029): 0.023*"#youtube" + 0.016*"#workout" + 0.015*"#fit" + 0.015*"#fitness" + 0.013*"#love" + 0.013*"#gym" + 0.012*"#health" +
0.012*"#fitfam" + 0.012*"#meditation" + 0.012*"#healthy"
2018-10-03 08:01:25,294 : INFO : topic #1 (0.040): 0.025*"#yogi" + 0.025*"#yoga" + 0.025*"#yogaeverydamnday" + 0.018*"#selfie" + 0.018*"thursday" + 0.017*"throwback" + 0.016
*"#yogaeverywhere" + 0.016*"#yogainspiration" + 0.009*"#yummy" + 0.009*"place"
2018-10-03 08:01:25,296 : INFO : topic #2 (0.030): 0.016*"night" + 0.013*"#olgod" + 0.008*"city" + 0.008*"amazing" + 0.008*"#friends" + 0.008*"real" + 0.007*"offer" + 0.007
*"cocktail" + 0.007*"beer" + 0.007*"shirt"
2018-10-03 08:01:25,298 : INFO : topic #3 (0.029): 0.035*"#graffiti" + 0.035*"#streetart" + 0.016*"#arteurbano" + 0.013*"#digerible" + 0.011*"#artecallejero" + 0.011*"#stree
tphotography" + 0.010*"#artederua" + 0.010*"arte" + 0.010*"urbano" + 0.008*"#arteenlacalle"
2018-10-03 08:01:25,301 : INFO : topic #4 (0.061): 0.031*"#sagradafamilia" + 0.029*"sagrada" + 0.019*"#travel" + 0.017*"#gaudi" + 0.015*"familia" + 0.013*"basílica" + 0.010
*"#love" + 0.010*"#photography" + 0.010*"#art" + 0.009*"#architecture"
2018-10-03 08:01:25,303 : INFO : topic #5 (0.036): 0.015*"#beach" + 0.012*"#photooftheday" + 0.011*"#friends" + 0.011*"#summer" + 0.011*"gothic" + 0.011*"#sagradafamilia" +
0.010*"quarter" + 0.008*"#amazing" + 0.008*"#pardes" + 0.008*"#smile"
2018-10-03 08:01:25,306 : INFO : topic #6 (0.039): 0.021*"#yum" + 0.014*"home" + 0.014*"place" + 0.013*"call" + 0.012*"yoga" + 0.012*"inner" + 0.012*"#yogabarcelona" + 0.012
*"#yogadiary" + 0.012*"#yogagirl" + 0.012*"#tapas"
```

Visualize Hashtag Pooling Model

In [61]:
```
pyLDAvis.enable_notebook()
vis_hp = pyLDAvis.gensim.prepare(lda_model_hashtag_pooling, corpus_hashtag_pooling, dictionary_hashtag_pooling)
```

In [62]:
```
vis_hp
```

Out[62]:

Intermediary Result:

After training ~50 models for each pooling method, the following conclusion was reached: An inspection of the topics of the no pooling and user pooling method shows that they are less human interpretable than hashtag pooling models and show some repetitions in words among topics. Moreover, the no pooling model and user pooling model are very unstable as tweets are very short. The best trained hashtag pooling model (meaning the one with the most human interpretable topics) will therefore be saved and used for further purposes. Hashtag pooling was also shown to give the best results in various research papers.

Save Models

In [49]:
```
lda_model_no_pooling.save('lda_model_no_pooling.model')
lda_model_user_pooling.save('lda_model_user_pooling.model')
lda_model_hashtag_pooling.save('lda_model_hashtag_pooling.model')
```

```
2018-10-03 08:31:58,659 : INFO : saving LdaState object under lda_model_no_pooling.model.state, separately None
2018-10-03 08:31:58,667 : INFO : saved lda_model_no_pooling.model.state
2018-10-03 08:31:58,675 : INFO : saving LdaModel object under lda_model_no_pooling.model, separately ['expElogbeta', 'sstats']
2018-10-03 08:31:58,678 : INFO : storing np array 'expElogbeta' to lda_model_no_pooling.model.expElogbeta.npy
2018-10-03 08:31:58,683 : INFO : not storing attribute dispatcher
2018-10-03 08:31:58,685 : INFO : not storing attribute id2word
2018-10-03 08:31:58,688 : INFO : not storing attribute state
2018-10-03 08:31:58,693 : INFO : saved lda_model_no_pooling.model
2018-10-03 08:31:58,695 : INFO : saving LdaState object under lda_model_user_pooling.model.state, separately None
2018-10-03 08:31:58,701 : INFO : saved lda_model_user_pooling.model.state
2018-10-03 08:31:58,708 : INFO : saving LdaModel object under lda_model_user_pooling.model, separately ['expElogbeta', 'sstats']
2018-10-03 08:31:58,711 : INFO : storing np array 'expElogbeta' to lda_model_user_pooling.model.expElogbeta.npy
2018-10-03 08:31:58,716 : INFO : not storing attribute dispatcher
2018-10-03 08:31:58,718 : INFO : not storing attribute id2word
2018-10-03 08:31:58,721 : INFO : not storing attribute state
2018-10-03 08:31:58,725 : INFO : saved lda_model_user_pooling.model
2018-10-03 08:31:58,727 : INFO : saving LdaState object under lda_model_hashtag_pooling.model.state, separately None
2018-10-03 08:31:58,735 : INFO : saved lda_model_hashtag_pooling.model.state
2018-10-03 08:31:58,744 : INFO : saving LdaModel object under lda_model_hashtag_pooling.model, separately ['expElogbeta', 'sstats']
2018-10-03 08:31:58,746 : INFO : storing np array 'expElogbeta' to lda_model_hashtag_pooling.model.expElogbeta.npy
2018-10-03 08:31:58,751 : INFO : not storing attribute dispatcher
2018-10-03 08:31:58,753 : INFO : not storing attribute id2word
2018-10-03 08:31:58,756 : INFO : not storing attribute state
2018-10-03 08:31:58,760 : INFO : saved lda_model_hashtag_pooling.model
```

Test Whether TFIDF Can Improve LDA (Instead of BOW)

Sometimes TFIDF improves LDA performance although LDA is mathematically meant to process a BOW input. TFIDF is therefore used to transform the corpus of the chosen model (hashtag pooling model with 7 topics).

In [73]:
```
# initialize tfidf model
tfidf_hashtag_pooling = models.TfidfModel(corpus_hashtag_pooling)

# run term frequency inverse document frequency transformation
# (transform bag-of-words integer counts corpus to tfidf real-valued weights
# corpus)
corpus_tfidf_hashtag_pooling = tfidf_hashtag_pooling[corpus_hashtag_pooling]
for doc in corpus_tfidf_hashtag_pooling:
    print(doc)
```

```
2018-10-02 07:22:21,349 : INFO : collecting document frequencies
2018-10-02 07:22:21,353 : INFO : PROGRESS: processing document #0
2018-10-02 07:22:21,478 : INFO : calculating IDF weights for 6040 documents and 9329 features (78970 matrix non-zeros)
```

```
[(0, 0.0199863227055571), (1, 0.039081489183023607), (2, 0.017151200283475224), (3, 0.031011931531736946), (4, 0.037735852213136988), (5, 0.01998632270555
7272708203), (7, 0.019154767272708203), (8, 0.014502628790419267), (9, 0.021158334261941181), (10, 0.017151200283475224), (11, 0.010123098737364748), (12,
58), (13, 0.019540744591511804), (14, 0.01497239563728543), (15, 0.0205133297581182), (16, 0.02316190125117416), (17, 0.018509762768885222), (18, 0.01856
9, 0.01636516604589474), (20, 0.019540744591511804), (21, 0.016810744159940039), (22, 0.013635165614322983), (23, 0.017337751212501114), (24, 0.03507435526
958377454182285), (26, 0.014162172666884083), (27, 0.0199863227055571), (28, 0.018509762768885222), (29, 0.017751389375493 06), (30, 0.015239404054736179),
4790078), (32, 0.0199863227055571), (33, 0.017537177602278825), (34, 0.017751389375493 06), (35, 0.023161901251174 16), (36, 0.017537177602278825), (37, 0.0
(38, 0.023708114594726614), (39, 0.030864355768496721), (40, 0.018118026174287 7208), (41, 0.017337751212501114), (42, 0.0205133297581182), (43, 0.018814311
19540744591511804), (45, 0.0205133297581182), (46, 0.016810744159940039), (47, 0.017751389375493 06), (48, 0.017751389375493 06), (49, 0.010860898832807251
72708203), (51, 0.019154767272708203), (52, 0.017537177602278825), (53, 0.019540744591511804), (54, 0.013857624286596286), (55, 0.057725015603421512), (56
804), (57, 0.021158334261941181), (58, 0.021158334261941181), (59, 0.019154767272708203), (60, 0.015334184223268162), (61, 0.021989889694790078), (62, 0.0
(63, 0.018509762768885222), (64, 0.0205133297581182), (65, 0.013582126765507704), (66, 0.029944791265457085), (67, 0.0199863227055571), (68, 0.02051332975
58334261941181), (70, 0.018509762768885222), (71, 0.0199863227055571), (72, 0.021158334261941181), (73, 0.041026659516236401), (74, 0.039972645411114199),
455765), (76, 0.019154767272708203), (77, 0.030668368446536323), (78, 0.018509762768885222), (79, 0.040746380296523 11), (80, 0.148078102151081 78), (81, 0.
8), (82, 0.019154767272708203), (83, 0.017337751212501114), (84, 0.018814311149173018), (85, 0.019154767272708203), (86, 0.123079978548709 21), (87, 0.02198
```

In [74]:
```
# train hashtag pooling model with tfidf corpus
lda_model_hashtag_pooling_tfidf = models.LdaModel(corpus_tfidf_hashtag_pooling,
                                        id2word=dictionary_hashtag_pooling,
                                        alpha='auto', eta='auto',
                                        eval_every=1,
                                        iterations=400, passes=20, num_topics=7)
```

```
2018-10-02 07:24:37,276 : INFO : using autotuned alpha, starting with [0.14285715, 0.14285715, 0.14285715, 0.14285715, 0.14285715, 0.14285715, 0.1428571
2018-10-02 07:24:37,280 : INFO : using serial LDA version on this node
2018-10-02 07:24:37,293 : INFO : running online (multi-pass) LDA training, 7 topics, 20 passes over the supplied corpus of 6040 documents, updating model
uments, evaluating perplexity every 2000 documents, iterating 400x with a convergence threshold of 0.001000
2018-10-02 07:24:38,243 : INFO : -13.890 per-word bound, 15180.5 perplexity estimate based on a held-out corpus of 2000 documents with 5704 words
2018-10-02 07:24:38,244 : INFO : PROGRESS: pass 0, at document #2000/6040
2018-10-02 07:24:38,884 : INFO : optimized alpha [0.12988012, 0.12784421, 0.12707655, 0.13058132, 0.1280534, 0.13039352, 0.12741488]
2018-10-02 07:24:38,886 : INFO : merging changes from 2000 documents into a model of 6040 documents
2018-10-02 07:24:38,896 : INFO : topic #2 (0.127): 0.003*"sagrada" + 0.002*"familia" + 0.002*"#sagradafamilia" + 0.002*"#illustration" + 0.002*"#catalonia
0.002*"#art" + 0.002*"best" + 0.002*"#gaudi" + 0.002*"#love"
2018-10-02 07:24:38,897 : INFO : topic #6 (0.127): 0.004*"gothic" + 0.003*"quarter" + 0.003*"#graffiti" + 0.003*"#sagradafamilia" + 0.002*"#beach" + 0.002
*"#arturba" + 0.002*"#selfie" + 0.002*"#bw" + 0.002*"#opium"
2018-10-02 07:24:38,900 : INFO : topic #0 (0.130): 0.005*"#sagradafamilia" + 0.005*"sagrada" + 0.004*"familia" + 0.003*"#catalonia" + 0.003*"#art" + 0.002
02*"gothic" + 0.002*"#friends" + 0.002*"going" + 0.002*"#beach"
2018-10-02 07:24:38,902 : INFO : topic #5 (0.130): 0.005*"sagrada" + 0.004*"família" + 0.003*"basílica" + 0.003*"familia" + 0.002*"#women" + 0.002*"#vacat
adafamilia" + 0.002*"#spoonie" + 0.002*"#neversettle" + 0.002*"#sport"
2018-10-02 07:24:38,905 : INFO : topic #3 (0.131): 0.005*"gothic" + 0.004*"quarter" + 0.004*"sagrada" + 0.003*"familia" + 0.002*"#sagradafamilia" + 0.002
*"#architecture" + 0.002*"love" + 0.002*"#playa" + 0.002*"#summer"
```

In [75]:
```
# print topics of model
_ = lda_model_hashtag_pooling_tfidf.print_topics()
```

```
2018-10-02 07:25:49,222 : INFO : topic #0 (0.055): 0.010*"inner" + 0.010*"#yogagirl" + 0.010*"#yogadiary" + 0.010*"#yogabarcelona" + 0.009*"yoga" + 0.008*"call" + 0.007*"pla
ce" + 0.007*"home" + 0.006*"nearly" + 0.006*"#sagradafamilia"
2018-10-02 07:25:49,225 : INFO : topic #1 (0.041): 0.005*"#yummy" + 0.004*"#youcanseeourhousefromhere" + 0.004*"#blueskies" + 0.004*"#watchcat" + 0.004*"chepolohelados" + 0.
004*"#youaresosocool" + 0.004*"meant" + 0.004*"you#bonepartyof" + 0.004*"perch" + 0.004*"pau"
2018-10-02 07:25:49,227 : INFO : topic #2 (0.040): 0.008*"arm" + 0.006*"sagrada" + 0.005*"#pardes" + 0.005*"família" + 0.005*"basílica" + 0.004*"#friends" + 0.003*"#liveasan
ative" + 0.003*"net" + 0.003*"revenue" + 0.003*"#yurbban"
2018-10-02 07:25:49,229 : INFO : topic #3 (0.036): 0.010*"#yum" + 0.007*"#youknowimright" + 0.007*"wife" + 0.006*"anyone" + 0.006*"#png" + 0.006*"#zoo" + 0.005*"zoo" + 0.005
*"#tapas" + 0.005*"beautiful" + 0.005*"tapa"
2018-10-02 07:25:49,231 : INFO : topic #4 (0.043): 0.008*"#youtube" + 0.005*"#youtubers" + 0.005*"#newvideo" + 0.005*"#pllfinale" + 0.005*"yayyy" + 0.005*"vídeo" + 0.005*"#t
easer" + 0.005*"nuevo" + 0.005*"#video" + 0.004*"zoo"
2018-10-02 07:25:49,235 : INFO : topic #5 (0.046): 0.016*"#yogaeverydamnday" + 0.016*"#yoga" + 0.013*"#yogainspiration" + 0.013*"#yogaeverywhere" + 0.012*"#yogi" + 0.010*"th
rowback" + 0.010*"thursday" + 0.008*"#selfie" + 0.005*"anyone#fitness" + 0.005*"#yogalove"
2018-10-02 07:25:49,237 : INFO : topic #6 (0.039): 0.005*"#yummi" + 0.005*"deal" + 0.005*"#nightout" + 0.005*"#hostallife" + 0.005*"#zarox" + 0.005*"cocktail" + 0.004*"#sagr
adafamilia" + 0.004*"#bookstagram" + 0.004*"tomorrow's" + 0.004*"#younglearners"
```

In the case of tweets, however, TFIDF does not improve the results but makes them worse and less interpretable. Very rare terms are weighted heavier but in the case of tweets these seldomly have an interpretable topic (e.g. "#youcanseeourhousefromhere"). The model that will be used as final LDA model is thus the 7 topics hashtag pooling model applied to a BOW corpus.

Analysis of Topics

In [77]:
```
pyLDAvis.enable_notebook()
vis_hp = pyLDAvis.gensim.prepare(lda_model_hashtag_pooling, corpus_hashtag_pooling, dictionary_hashtag_pooling)
```

In [78]:
```
vis_hp
```

Out[78]:

Manual inspection of the topics leads to the following labels:

Topic 0: Sightseeing (Sagrada Familia, gaudi, architecture, travel, church ...)

Topic 1: Summer, Sun & Friends (beach, friends, summer. smile, sun...)

Topic 2: Streetart (graffiti, streetart, arte urbano, massive, streetphotography...)

Topic 3: Everyday Life (yum, home, place, call, tapas ...)

Topic 4: Lifestyle & Culture (yoga, selfie, contemporaryart, yummy, brianeno ...)

Topic 5: Nightlife (night, olgod beer bar, cocktail, beer, raval ...)

Topic 6: Sports, Health & Image (workout, fit, meditation, healthy, video ...)

In [79]:
```
# display the 10 most important words for each topic
n_topics = 7
topic_terms = []

for i in range(0, n_topics):
    temp = lda_model_hashtag_pooling.show_topic(i, 10)
    terms = []
    for term in temp:
        terms.append(term)
    topic_terms.append(terms)
    print("Top 10 terms for topic #" + str(i) + ": "+ ", ".join([str(i[0]) for i in terms]))
```

```
Top 10 terms for topic #0: #youtube, #workout, #fit, #fitness, #love, #gym, #health, #fitfam, #meditation, #healthy
Top 10 terms for topic #1: #yogi, #yoga, #yogaeverydamnday, #selfie, thursday, throwback, #yogaeverywhere, #yogainspiration, #yummy, place
Top 10 terms for topic #2: night, #ølgod, city, amazing, #friends, real, offer, cocktail, beer, shirt
Top 10 terms for topic #3: #graffiti, #streetart, #arteurbano, #digerible, #artecallejero, #streetphotography, #artederua, arte, urbano, #arteenlacalle
Top 10 terms for topic #4: #sagradafamilia, sagrada, #travel, #gaudi, familia, basílica, #love, #photography, #art, #architecture
Top 10 terms for topic #5: #beach, #photooftheday, #friends, #summer, gothic, #sagradafamilia, quarter, #amazing, #pardes, #smile
Top 10 terms for topic #6: #yum, home, place, call, yoga, inner, #yogabarcelona, #yogadiary, #yogagirl, #tapas
```

In [80]:
```
# display wordclouds for the topics
def terms_to_wordcounts(terms, multiplier=1000):
    return  " ".join([" ".join(int(multiplier*i[1]) * [i[0]]) for i in terms])

wordclouds = []
i = 0

for topic in topic_terms:
    wordcloud = WordCloud(background_color="black", collocations=False).generate(terms_to_wordcounts(topic))

    _ = plt.imshow(wordcloud)
    _ = plt.axis("off")
    _ = plt.savefig("terms_wordcloud_topic" + str(i))
    _ = plt.show()

    i += 1
```

Jupyter Notebook 3: Train NMF models

References

In [1]:
```
# https://medium.com/mlreview/topic-modeling-with-scikit-learn-e80d33668730
# http://scikit-learn.org/stable/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html
# https://github.com/derekgreene/topic-model-tutorial/blob/master/3%20-%20Parameter%20Selection%20for%20NMF.ipynb
```

Prepare Notebook

In [2]:
```
# import packages
from sklearn.decomposition import NMF
import os
from gensim import corpora
from sklearn.feature_extraction.text import TfidfVectorizer
import pickle
import gensim.models
import numpy as np
from gensim.models import Word2Vec
from sklearn import decomposition
import gensim.models
from itertools import combinations
import matplotlib
import matplotlib.pyplot as plt
```

```
C:\Users\Sebastian Birk\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chunkize to c
hunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [3]:
```
# display graphs in jupyter
%matplotlib inline
```

Load Vectorized Corpora and Dictionaries

In [4]:
```
# load no pooling corpus
if (os.path.exists("tourism_no_pooling.dict")):
   dictionary_no_pooling = corpora.Dictionary.load('tourism_no_pooling.dict')
   corpus_no_pooling = corpora.MmCorpus('tourism_no_pooling.mm')
   print("Vectorized no pooling corpus loaded!")
else:
   print("Please run preprocessing script first!")
```

```
Vectorized no pooling corpus loaded!
```

Load Documents and Unpreprocessed Tokenized Documents

In [5]:
```
# load no pooling documents
with open ('nmf_documents_no_pooling.p', 'rb') as fp:
    documents_no_pooling = pickle.load(fp)

# load no pooling unpreprocessed tokenized documents
with open ('tokenized_documents_no_pooling_unpp.p', 'rb') as fp:
    tokenized_documents_no_pooling = pickle.load(fp)
```

Apply TFIDF

In [6]:
```
# tfidf is usually used before running NMF
tfidf_vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, stop_words='english')
tfidf_no_pooling = tfidf_vectorizer.fit_transform(documents_no_pooling)
tfidf_feature_names_no_pooling = tfidf_vectorizer.get_feature_names()
```

Run NMF Model and Determine Number of Topics

In [7]:
```
# define range of possible topic numbers
kmin, kmax = 4, 8
```

In [8]:
```
# run NMF model for different number of topics
topic_models = []
# try each value of k
for k in range(kmin,kmax+1):
    print("Applying NMF for k=%d ..." % k)
    # run NMF
    model = decomposition.NMF( init="nndsvd", n_components=k)
    W = model.fit_transform(tfidf_no_pooling)
    H = model.components_
    # store for later
    topic_models.append((k,W,H))
```

```
Applying NMF for k=4 ...
Applying NMF for k=5 ...
Applying NMF for k=6 ...
Applying NMF for k=7 ...
Applying NMF for k=8 ...
```

In [9]:
```
# prepare tokenized documents
tokenized_documents_no_pooling = [[item.replace("#","") for item in document] for document in tokenized_documents_no_pooling]
```

In [10]:
```
# display tokenized documents
tokenized_documents_no_pooling
```

Out[10]:
```
[['brew',
  'pub',
  'to',
  'try',
  'a',
  'few',
  'of',
  'the',
  'beers',
  'on',
  'offer',
  'olgodbarcelona',
  'ølgod',
  'triathlontraining'],
 ['art',
  'is',
  'coming',
  'face',
  'to',
```

In [11]:
```
# build a word2vec model
w2v_model = gensim.models.Word2Vec(tokenized_documents_no_pooling, min_count=2, size=500)
```

In [12]:
```
# define function to calculate coherence
def calculate_coherence(w2v_model, term_rankings):
    overall_coherence = 0.0
    for topic_index in range(len(term_rankings)):
        # check each pair of terms
        pair_scores = []
        for pair in combinations(term_rankings[topic_index], 2):
            pair_scores.append(w2v_model.similarity(pair[0], pair[1]) )
        # get the mean for all pairs in this topic
        topic_score = sum(pair_scores) / len(pair_scores)
        overall_coherence += topic_score
    # get the mean score across all topics
    return overall_coherence / len(term_rankings)
```

In [13]:
```
# define function to get top terms
def get_descriptor(all_terms, H, topic_index, top):
    # reverse sort the values to sort the indices
    top_indices = np.argsort(H[topic_index,:] )[::-1]
    # now get the terms corresponding to the top-ranked indices
    top_terms = []
    for term_index in top_indices[0:top]:
        top_terms.append(all_terms[term_index])
    return top_terms
```

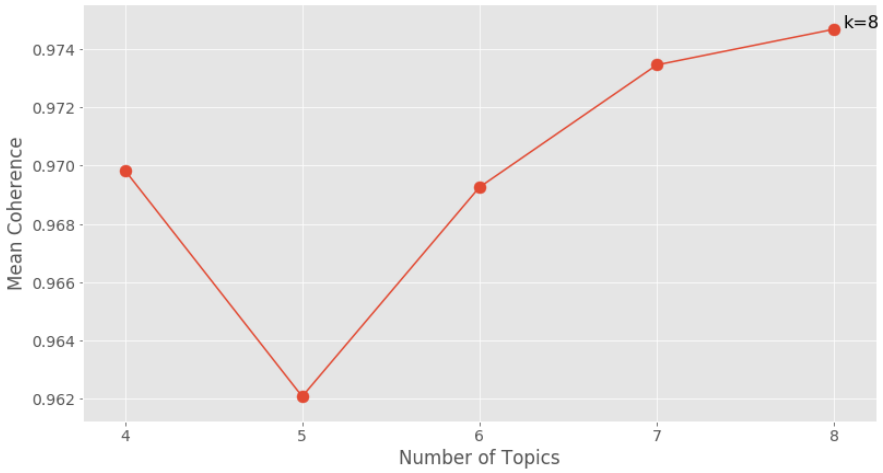
In [14]:
```
# run functions
k_values = []
coherences = []
for (k,W,H) in topic_models:
    # Get all of the topic descriptors - the term_rankings, based on top 10 terms
    term_rankings = []
    for topic_index in range(k):
        term_rankings.append(get_descriptor(tfidf_feature_names_no_pooling, H, topic_index, 10))
    # Now calculate the coherence based on our Word2vec model
    k_values.append( k )
    coherences.append(calculate_coherence(w2v_model,term_rankings))
    print("K=%02d: Coherence=%.4f" % ( k, coherences[-1] ) )
```

```
K=04: Coherence=0.9698
K=05: Coherence=0.9621
K=06: Coherence=0.9693
K=07: Coherence=0.9735
K=08: Coherence=0.9747
```

```
C:\Users\Sebastian Birk\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DeprecationWarning: Call to deprecated `similarity
` (Method will be removed in 4.0.0, use self.wv.similarity() instead).
```

In [15]:
```
# graph settings
plt.style.use("ggplot")
matplotlib.rcParams.update({"font.size": 14})
```

In [17]:
```
# create graph
fig = plt.figure(figsize=(13,7))
# create the line plot
ax = plt.plot( k_values, coherences )
plt.xticks(k_values)
plt.xlabel("Number of Topics")
plt.ylabel("Mean Coherence")
# add the points
plt.scatter( k_values, coherences, s=120)
# find and annotate the maximum point on the plot
ymax = max(coherences)
xpos = coherences.index(ymax)
best_k = k_values[xpos]
plt.annotate( "k=%d" % best_k, xy=(best_k, ymax), xytext=(best_k, ymax), textcoords="offset points", fontsize=16)
plt.savefig("coherence_scores_nmf")
# show the plot
plt.show()
```



In [18]:
```
# select the best number of topics according to tc-w2v coherence measure
k = best_k
# get the model that we generated earlier.
W = topic_models[k-kmin][1]
H = topic_models[k-kmin][2]
```

In [19]:
```
# print the top words for each topic
for topic_index in range(k):
    descriptor = get_descriptor(tfidf_feature_names_no_pooling, H, topic_index, 20)
    str_descriptor = ", ".join( descriptor )
    print("Topic %02d: %s" % ( topic_index+1, str_descriptor ) )
```

```
Topic 01: familia, sagrada, la, basilica, inside, ll, construction, beautiful, dance, view, beauty, time, good, antoni, video,
incredible, capture, night, família, years
Topic 02: gothic, quarter, gothicquarter, art, streets, barcelona, graffiti, street, streetart, lost, travel, night, strolling,
today, beautiful, love, life, bcn, favorite, time
Topic 03: just, ramblas, parc, laberint, video, arenas, aquàrium, platja, avinguda, diagonal, plaza, barceloneta, joan, guell,
fundació, miró, llevant, zoo, serras, princess
Topic 04: basílica, família, sagrada, beautiful, lasagradafamilia, barcelona, la, time, inside, expiatori, place, beauty, templ
e, church, finished, light, gaudí, looking, look, dream
Topic 05: sagradafamilia, gaudi, architecture, travel, church, amazing, beautiful, basilica, barcelona, gaudí, catalonia, love,
expiatorio, templo, building, holiday, masterpiece, inside, tourist, art
Topic 06: milà, pedreracasa, casa, pedrera, gaudi, gaudí, lapedrera, history, barcelona, beautiful, casamila, art, photography,
weekend, architecture, great, roof, real, rooftop, nature
Topic 07: beach, city, marbella, barceloneta, love, playa, festival, barcelona, summer, time, beautiful, platja, sun, travel, n
ight, friends, bbf, happy, cruïlla, best
Topic 08: drinking, photo, cervecita, olgodbeer, rosses, torrades, ipa, rovira, cocovailbh, edgebrewing, craft, cuc, cal, omnip
ollo, garagebeerco, birrificio, vibrantforest, ale, lambrate, foundersbrewing
```

Upon inspection, NMF works very well (seems to be more interpretable than the LDA no pooling model). However, the topics are very repetitive and not as clearly interpretable and differentiatable as the topcis detected by the LDA model with hashtag pooling. The problem of repetitions of keywords among different topics complicates the matter.

Jupyter Notebook 4: Train HDP & train LSI models

References

In [1]:
```
# https://markroxor.github.io/gensim/static/notebooks/gensim_news_classification.html
# https://medium.com/square-corner-blog/topic-modeling-optimizing-for-human-interpretability-48a81f6ce0ed
```

Prepare Notebook

In [2]:
```
# import packages
import logging
import os
import pickle
import numpy as np
import matplotlib.pyplot as plt
from gensim import corpora, models
from gensim.models import HdpModel, LsiModel, CoherenceModel
```

```
C:\Users\Sebastian Birk\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chunkize to c
hunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [3]:
```
# log events
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

Load Training Corpora and Dictionaries

In [4]:
```
# load no pooling corpus
if (os.path.exists("tourism_no_pooling.dict")):
   dictionary_no_pooling = corpora.Dictionary.load('tourism_no_pooling.dict')
   corpus_no_pooling = corpora.MmCorpus('tourism_no_pooling.mm')
   print("Vectorized no pooling corpus loaded!")
else:
   print("Please run preprocessing script first!")

# load user pooling corpus
if (os.path.exists("tourism_user_pooling.dict")):
   dictionary_user_pooling = corpora.Dictionary.load('tourism_user_pooling.dict')
   corpus_user_pooling = corpora.MmCorpus('tourism_user_pooling.mm')
   print("Vectorized user pooling corpus loaded!")
else:
   print("Please run preprocessing script first!")

# load hashtag pooling corpus
if (os.path.exists("tourism_hashtag_pooling.dict")):
   dictionary_hashtag_pooling = corpora.Dictionary.load('tourism_hashtag_pooling.dict')
   corpus_hashtag_pooling = corpora.MmCorpus('tourism_hashtag_pooling.mm')
   print("Vectorized hashtag pooling corpus loaded!")
else:
   print("Please run preprocessing script first!")
```

```
2018-10-03 08:47:40,731 : INFO : loading Dictionary object from tourism_no_pooling.dict
2018-10-03 08:47:40,740 : INFO : loaded tourism_no_pooling.dict
2018-10-03 08:47:40,745 : INFO : loaded corpus index from tourism_no_pooling.mm.index
2018-10-03 08:47:40,747 : INFO : initializing cython corpus reader from tourism_no_pooling.mm
2018-10-03 08:47:40,751 : INFO : accepted corpus with 7633 documents, 4790 features, 37016 non-zero entries
2018-10-03 08:47:40,753 : INFO : loading Dictionary object from tourism_user_pooling.dict
2018-10-03 08:47:40,759 : INFO : loaded tourism_user_pooling.dict
2018-10-03 08:47:40,763 : INFO : loaded corpus index from tourism_user_pooling.mm.index
2018-10-03 08:47:40,764 : INFO : initializing cython corpus reader from tourism_user_pooling.mm
2018-10-03 08:47:40,768 : INFO : accepted corpus with 4424 documents, 3990 features, 30119 non-zero entries
2018-10-03 08:47:40,770 : INFO : loading Dictionary object from tourism_hashtag_pooling.dict
2018-10-03 08:47:40,779 : INFO : loaded tourism_hashtag_pooling.dict
2018-10-03 08:47:40,783 : INFO : loaded corpus index from tourism_hashtag_pooling.mm.index
2018-10-03 08:47:40,785 : INFO : initializing cython corpus reader from tourism_hashtag_pooling.mm
2018-10-03 08:47:40,789 : INFO : accepted corpus with 6040 documents, 9330 features, 78970 non-zero entries
```

```
Vectorized no pooling corpus loaded!
Vectorized user pooling corpus loaded!
Vectorized hashtag pooling corpus loaded!
```

Load Documents

In [5]:
```
# load no pooling documents
with open ('tokenized_documents_no_pooling.p', 'rb') as fp:
    tokenized_documents_no_pooling = pickle.load(fp)

# load user pooling documents
with open('tokenized_documents_user_pooling.p', 'rb') as fp:
    tokenized_documents_user_pooling = pickle.load(fp)

# load hashtag pooling documents
with open('tokenized_documents_hashtag_pooling.p', 'rb') as fp:
    tokenized_documents_hashtag_pooling = pickle.load(fp)
```

Load LDA Models (Trained in TopicModeling#2 Script)

In [6]:
```
# load models
lda_model_no_pooling = models.LdaModel.load('lda_model_no_pooling.model') # 6 topics
lda_model_user_pooling = models.LdaModel.load('lda_model_user_pooling.model') # 7 topics
lda_model_hashtag_pooling = models.LdaModel.load('lda_model_hashtag_pooling.model') # 7 topics
```

```
2018-10-03 08:47:44,233 : INFO : loading LdaModel object from lda_model_no_pooling.model
2018-10-03 08:47:44,238 : INFO : loading expElogbeta from lda_model_no_pooling.model.expElogbeta.npy with mmap=None
2018-10-03 08:47:44,241 : INFO : setting ignored attribute dispatcher to None
2018-10-03 08:47:44,243 : INFO : setting ignored attribute id2word to None
2018-10-03 08:47:44,246 : INFO : setting ignored attribute state to None
2018-10-03 08:47:44,247 : INFO : loaded lda_model_no_pooling.model
2018-10-03 08:47:44,249 : INFO : loading LdaState object from lda_model_no_pooling.model.state
2018-10-03 08:47:44,253 : INFO : loaded lda_model_no_pooling.model.state
2018-10-03 08:47:44,261 : INFO : loading LdaModel object from lda_model_user_pooling.model
2018-10-03 08:47:44,264 : INFO : loading expElogbeta from lda_model_user_pooling.model.expElogbeta.npy with mmap=None
2018-10-03 08:47:44,267 : INFO : setting ignored attribute dispatcher to None
2018-10-03 08:47:44,269 : INFO : setting ignored attribute id2word to None
2018-10-03 08:47:44,270 : INFO : setting ignored attribute state to None
2018-10-03 08:47:44,272 : INFO : loaded lda_model_user_pooling.model
2018-10-03 08:47:44,274 : INFO : loading LdaState object from lda_model_user_pooling.model.state
2018-10-03 08:47:44,278 : INFO : loaded lda_model_user_pooling.model.state
2018-10-03 08:47:44,285 : INFO : loading LdaModel object from lda_model_hashtag_pooling.model
2018-10-03 08:47:44,288 : INFO : loading expElogbeta from lda_model_hashtag_pooling.model.expElogbeta.npy with mmap=None
2018-10-03 08:47:44,291 : INFO : setting ignored attribute dispatcher to None
2018-10-03 08:47:44,294 : INFO : setting ignored attribute id2word to None
2018-10-03 08:47:44,297 : INFO : setting ignored attribute state to None
2018-10-03 08:47:44,302 : INFO : loaded lda_model_hashtag_pooling.model
2018-10-03 08:47:44,308 : INFO : loading LdaState object from lda_model_hashtag_pooling.model.state
2018-10-03 08:47:44,317 : INFO : loaded lda_model_hashtag_pooling.model.state
```

In [7]:
```
# extract topics with word probabilities
lda_topics_no_pooling = lda_model_no_pooling.show_topics(formatted=False)
lda_topics_user_pooling = lda_model_user_pooling.show_topics(formatted=False)
lda_topics_hashtag_pooling = lda_model_hashtag_pooling.show_topics(formatted=False)
```

Train HDP Models

In [8]:
```
# train HDP models with different pooling methods (similar to LDA in TopicModeling#2 Script)
hdp_model_no_pooling = HdpModel(corpus_no_pooling, dictionary_no_pooling)
hdp_model_user_pooling = HdpModel(corpus_user_pooling, dictionary_user_pooling)
hdp_model_hashtag_pooling = HdpModel(corpus_hashtag_pooling, dictionary_hashtag_pooling)
```

```
2018-10-03 08:48:10,608 : INFO : (0, '0.001*building + 0.001*#historicalplace + 0.001*#barceloneando + 0.001*soooo + 0.001*o
pen + 0.001*guesthouse + 0.001*menu + 0.001*legend + 0.001*#tomato + 0.001*milkandcoworking')
2018-10-03 08:48:10,614 : INFO : (1, '0.002*solstice + 0.002*moment + 0.002*recipe + 0.002*#puro + 0.002*cruilla + 0.002*tim
e + 0.001*hotel + 0.001*father + 0.001*cold + 0.001*traveled')
2018-10-03 08:48:10,626 : INFO : (2, '0.002*baby + 0.002*shutter + 0.002*fòrum + 0.001*cn + 0.001*#sagrada + 0.001*shortest
+ 0.001*velvet + 0.001*#moments + 0.001*everywhere + 0.001*sagrada')
2018-10-03 08:48:10,632 : INFO : (3, '0.002*time + 0.002*tradition + 0.002*cabaret + 0.001*uniqlo + 0.001*#edm + 0.001*#curi
ocollection + 0.001*#fashionblogger + 0.001*#train + 0.001*taste + 0.001*espectacular')
2018-10-03 08:48:10,639 : INFO : (4, '0.002*oficial + 0.002*#tattoo + 0.001*said + 0.001*#maddydoeseurope + 0.001*imaginatio
n + 0.001*alberto + 0.001*went + 0.001*#birthday + 0.001*exists + 0.001*fabrica')
2018-10-03 08:48:10,657 : INFO : (5, '0.002*va + 0.002*tempura + 0.002*pecan + 0.001*mile + 0.001*cannot + 0.001*#rockandrol
l + 0.001*#drummer + 0.001*start + 0.001*future + 0.001*montse')
2018-10-03 08:48:10,669 : INFO : (6, '0.002*robb + 0.002*drawing + 0.001*amb + 0.001*#temple + 0.001*soo + 0.001*late + 0.00
1*#amsterdam + 0.001*sustainable + 0.001*#sightsee + 0.001*site')
2018-10-03 08:48:10,682 : INFO : (7, '0.002*#chimney + 0.002*sagrada + 0.001*dessert + 0.001*vigo + 0.001*frank + 0.001*#end
ofsummer + 0.001*#architect + 0.001*gave + 0.001*takeover + 0.001*#tan')
2018-10-03 08:48:10,697 : INFO : (8, '0.002*margot + 0.002*#basilicadelasagradafamilia + 0.002*boutique + 0.002*bird + 0.002
*away + 0.002*#watercolorpencil + 0.001*express + 0.001*#usa + 0.001*ironhack + 0.001*santa')
2018-10-03 08:48:10,703 : INFO : (9, '0.002*churchill + 0.002*petit + 0.001*#shopping + 0.001*progress + 0.001*#color + 0.00
```

In [9]:
```
# exctract topics with word probabilities
hdp_topics_no_pooling = hdp_model_no_pooling.show_topics(formatted=False)
hdp_topics_user_pooling = hdp_model_user_pooling.show_topics(formatted=False)
hdp_topics_hashtag_pooling = hdp_model_hashtag_pooling.show_topics(formatted=False)
```

Train LSI Models

In [10]:
```
# train LSI models with similar configurations as LDA models
lsi_model_no_pooling = LsiModel(corpus=corpus_no_pooling, num_topics=6, id2word=dictionary_no_pooling)
lsi_model_user_pooling = LsiModel(corpus=corpus_user_pooling, num_topics=7, id2word=dictionary_user_pooling)
lsi_model_hashtag_pooling = LsiModel(corpus=corpus_hashtag_pooling, num_topics=7, id2word=dictionary_hashtag_pooling)
```

```
2018-10-03 08:48:52,841 : INFO : using serial LSI version on this node
2018-10-03 08:48:52,845 : INFO : updating model with new documents
2018-10-03 08:48:52,896 : INFO : preparing a new chunk of documents
2018-10-03 08:48:52,927 : INFO : using 100 extra samples and 2 power iterations
2018-10-03 08:48:52,929 : INFO : 1st phase: constructing (4790, 106) action matrix
2018-10-03 08:48:52,986 : INFO : orthonormalizing (4790, 106) action matrix
2018-10-03 08:48:53,145 : INFO : 2nd phase: running dense svd on (106, 7633) matrix
2018-10-03 08:48:53,267 : INFO : computing the final decomposition
2018-10-03 08:48:53,269 : INFO : keeping 6 factors (discarding 64.808% of energy spectrum)
2018-10-03 08:48:53,274 : INFO : processed documents up to #7633
2018-10-03 08:48:53,278 : INFO : topic #0(48.355): -0.783*"sagrada" + -0.533*"familia" + -0.213*"basílica" + -0.189*"família" +
-0.103*"#sagradafamilia" + -0.033*"#gaudi" + -0.032*"beautiful" + -0.027*"gaudi" + -0.024*"inside" + -0.022*"basilica"
2018-10-03 08:48:53,280 : INFO : topic #1(28.253): 0.728*"gothic" + 0.676*"quarter" + 0.034*"street" + 0.032*"#art" + 0.028*"#g
othicquarter" + 0.028*"#streetart" + 0.025*"#graffiti" + 0.025*"night" + 0.024*"love" + 0.024*"#travel"
2018-10-03 08:48:53,283 : INFO : topic #2(25.656): 0.611*"basílica" + -0.555*"familia" + 0.548*"família" + 0.068*"#sagradafamil
ia" + 0.063*"sagrada" + 0.039*"beautiful" + 0.020*"place" + 0.020*"time" + 0.019*"#catalonia" + 0.018*"#lasagradafamilia"
2018-10-03 08:48:53,287 : INFO : topic #3(20.179): -0.898*"#sagradafamilia" + -0.278*"#gaudi" + 0.128*"família" + -0.124*"#trav
el" + -0.106*"#architecture" + 0.096*"sagrada" + -0.059*"#summer" + -0.059*"#church" + -0.057*"#catalonia" + -0.054*"#basilica"
2018-10-03 08:48:53,290 : INFO : topic #4(16.471): -0.429*"#streetart" + -0.347*"drinking" + -0.304*"#arteurbano" + -0.280*"#di
gerible" + -0.269*"#graffiti" + -0.238*"time" + -0.225*"arte" + -0.223*"urbano" + -0.178*"beach" + -0.163*"tres"
2018-10-03 08:48:53,298 : INFO : using serial LSI version on this node
2018-10-03 08:48:53,303 : INFO : updating model with new documents
2018-10-03 08:48:53,359 : INFO : preparing a new chunk of documents
2018-10-03 08:48:53,385 : INFO : using 100 extra samples and 2 power iterations
2018-10-03 08:48:53,387 : INFO : 1st phase: constructing (3990, 107) action matrix
2018-10-03 08:48:53,435 : INFO : orthonormalizing (3990, 107) action matrix
2018-10-03 08:48:53,562 : INFO : 2nd phase: running dense svd on (107, 4424) matrix
2018-10-03 08:48:53,641 : INFO : computing the final decomposition
2018-10-03 08:48:53,644 : INFO : keeping 7 factors (discarding 50.533% of energy spectrum)
2018-10-03 08:48:53,648 : INFO : processed documents up to #4424
2018-10-03 08:48:53,651 : INFO : topic #0(84.240): 0.523*"#streetart" + 0.511*"#arteurbano" + 0.404*"arte" + 0.278*"tres" + 0.2
77*"xemeneies" + 0.274*"#graffiti" + 0.242*"#arteenlacalle" + 0.058*"#instagood" + 0.050*"#artecallejero" + 0.043*"#streetphoto
graphy"
2018-10-03 08:48:53,654 : INFO : topic #1(65.004): -0.746*"#photodrinking" + -0.506*"cervecita" + -0.271*"drinking" + -0.145*"r
oss" + -0.145*"torrades" + -0.118*"cal" + -0.106*"ipa" + -0.077*"birrificio" + -0.066*"olgodbeer" + -0.060*"lambrate"
2018-10-03 08:48:53,656 : INFO : topic #2(57.917): -0.768*"sagrada" + -0.515*"familia" + -0.213*"basílica" + -0.184*"família" +
-0.138*"#sagradafamilia" + -0.084*"gothic" + -0.075*"quarter" + -0.052*"#gaudi" + -0.046*"beautiful" + -0.042*"gaudi"
2018-10-03 08:48:53,659 : INFO : topic #3(43.101): -0.442*"#graffiti" + -0.426*"#artecallejero" + -0.369*"#streetphotography" +
-0.308*"#catalonia" + -0.251*"#catalunya" + -0.232*"#tweegram" + 0.218*"#arteurbano" + -0.214*"#graffitiart" + 0.179*"arte" + -
0.165*"#streetart"
2018-10-03 08:48:53,662 : INFO : topic #4(41.794): 0.521*"tip" + 0.520*"insider" + 0.436*"experience" + 0.435*"food" + 0.108*"b
ar" + 0.105*"ticket" + 0.097*"travel" + 0.095*"dirty" + 0.095*"south" + 0.095*"#ticketsbar"
2018-10-03 08:48:53,668 : INFO : using serial LSI version on this node
2018-10-03 08:48:53,670 : INFO : updating model with new documents
2018-10-03 08:48:53,816 : INFO : preparing a new chunk of documents
2018-10-03 08:48:53,856 : INFO : using 100 extra samples and 2 power iterations
2018-10-03 08:48:53,858 : INFO : 1st phase: constructing (9330, 107) action matrix
2018-10-03 08:48:53,927 : INFO : orthonormalizing (9330, 107) action matrix
2018-10-03 08:48:54,222 : INFO : 2nd phase: running dense svd on (107, 6040) matrix
2018-10-03 08:48:54,321 : INFO : computing the final decomposition
2018-10-03 08:48:54,322 : INFO : keeping 7 factors (discarding 24.657% of energy spectrum)
2018-10-03 08:48:54,329 : INFO : processed documents up to #6040
2018-10-03 08:48:54,332 : INFO : topic #0(569.920): 0.686*"#sagradafamilia" + 0.417*"sagrada" + 0.255*"#gaudi" + 0.244*"famili
a" + 0.184*"#travel" + 0.181*"basílica" + 0.114*"#summer" + 0.111*"#architecture" + 0.075*"#catalonia" + 0.075*"#love"
2018-10-03 08:48:54,335 : INFO : topic #1(245.858): -0.544*"#streetart" + -0.410*"#graffiti" + -0.304*"#arteurbano" + -0.268*"#
digerible" + 0.222*"#sagradafamilia" + -0.199*"urbano" + -0.199*"arte" + -0.156*"#arteenlacalle" + -0.137*"tres" + -0.137*"xeme
neies"
2018-10-03 08:48:54,337 : INFO : topic #2(204.053): 0.541*"#travel" + -0.435*"#sagradafamilia" + 0.242*"#summer" + 0.215*"#lov
e" + -0.186*"#streetart" + 0.158*"#photography" + -0.157*"#arteurbano" + -0.148*"#digerible" + 0.129*"#beach" + 0.119*"gothic"
2018-10-03 08:48:54,340 : INFO : topic #3(150.729): 0.790*"#gaudi" + -0.277*"#summer" + -0.251*"#love" + 0.238*"#architecture"
+ -0.221*"#sagradafamilia" + -0.112*"#friends" + 0.109*"#lapedrera" + -0.097*"#beach" + -0.092*"#sun" + 0.056*"#casamila"
2018-10-03 08:48:54,342 : INFO : topic #4(144.553): 0.691*"#travel" + -0.438*"#summer" + -0.259*"#gaudi" + -0.164*"#love" + -0.
161*"#friends" + 0.156*"#sagradafamilia" + -0.119*"#beach" + -0.086*"#architecture" + -0.070*"#catalunya" + 0.064*"#photograph
y"
```

In [11]:
```
# extract topics with word probabilities
lsi_topics_no_pooling = lsi_model_no_pooling.show_topics(formatted=False)
lsi_topics_user_pooling = lsi_model_user_pooling.show_topics(formatted=False)
lsi_topics_hashtag_pooling = lsi_model_hashtag_pooling.show_topics(formatted=False)
```

Calculate Coherence Values

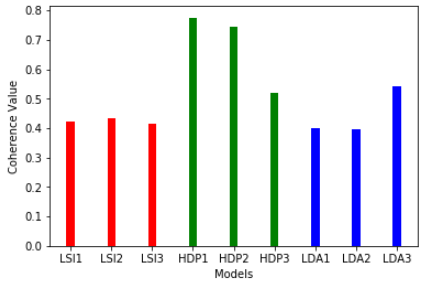
In [12]:
```
# extract top topic words
lsi_topics_no_pooling = [[word for word, prob in topic] for topicid, topic in lsi_topics_no_pooling]
lsi_topics_user_pooling = [[word for word, prob in topic] for topicid, topic in lsi_topics_user_pooling]
lsi_topics_hashtag_pooling = [[word for word, prob in topic] for topicid, topic in lsi_topics_hashtag_pooling]

hdp_topics_no_pooling = [[word for word, prob in topic] for topicid, topic in hdp_topics_no_pooling]
hdp_topics_user_pooling = [[word for word, prob in topic] for topicid, topic in hdp_topics_user_pooling]
hdp_topics_hashtag_pooling = [[word for word, prob in topic] for topicid, topic in hdp_topics_hashtag_pooling]

lda_topics_no_pooling = [[word for word, prob in topic] for topicid, topic in lda_topics_no_pooling]
lda_topics_user_pooling = [[word for word, prob in topic] for topicid, topic in lda_topics_user_pooling]
lda_topics_hashtag_pooling = [[word for word, prob in topic] for topicid, topic in lda_topics_hashtag_pooling]
```

In [13]:
```
# calculate coherences
lsi_no_pooling_coherence = CoherenceModel(topics=lsi_topics_no_pooling, texts=tokenized_documents_no_pooling, dictionary=dictiona
lsi_user_pooling_coherence = CoherenceModel(topics=lsi_topics_user_pooling, texts=tokenized_documents_user_pooling, dictionary=di
lsi_hashtag_pooling_coherence = CoherenceModel(topics=lsi_topics_hashtag_pooling, texts=tokenized_documents_hashtag_pooling, dict

hdp_no_pooling_coherence = CoherenceModel(topics=hdp_topics_no_pooling, texts=tokenized_documents_no_pooling, dictionary=dictiona
hdp_user_pooling_coherence = CoherenceModel(topics=hdp_topics_user_pooling, texts=tokenized_documents_user_pooling, dictionary=di
hdp_hashtag_pooling_coherence = CoherenceModel(topics=hdp_topics_hashtag_pooling, texts=tokenized_documents_hashtag_pooling, dict

lda_no_pooling_coherence = CoherenceModel(topics=lda_topics_no_pooling, texts=tokenized_documents_no_pooling, dictionary=dictiona
lda_user_pooling_coherence = CoherenceModel(topics=lda_topics_user_pooling, texts=tokenized_documents_user_pooling, dictionary=di
lda_hashtag_pooling_coherence = CoherenceModel(topics=lda_topics_hashtag_pooling, texts=tokenized_documents_hashtag_pooling, dict
```

```
2018-10-03 08:48:59,161 : INFO : 16 batches submitted to accumulate stats from 1024 documents (1824 virtual)
2018-10-03 08:48:59,168 : INFO : 17 batches submitted to accumulate stats from 1088 documents (1948 virtual)
2018-10-03 08:48:59,171 : INFO : 18 batches submitted to accumulate stats from 1152 documents (2063 virtual)
2018-10-03 08:48:59,179 : INFO : 19 batches submitted to accumulate stats from 1216 documents (2183 virtual)
2018-10-03 08:48:59,190 : INFO : 20 batches submitted to accumulate stats from 1280 documents (2296 virtual)
2018-10-03 08:48:59,196 : INFO : 21 batches submitted to accumulate stats from 1344 documents (2387 virtual)
2018-10-03 08:48:59,202 : INFO : 22 batches submitted to accumulate stats from 1408 documents (2416 virtual)
2018-10-03 08:48:59,208 : INFO : 23 batches submitted to accumulate stats from 1472 documents (2479 virtual)
2018-10-03 08:48:59,216 : INFO : 24 batches submitted to accumulate stats from 1536 documents (2538 virtual)
2018-10-03 08:48:59,219 : INFO : 25 batches submitted to accumulate stats from 1600 documents (2576 virtual)
2018-10-03 08:48:59,222 : INFO : 26 batches submitted to accumulate stats from 1664 documents (2627 virtual)
2018-10-03 08:48:59,231 : INFO : 27 batches submitted to accumulate stats from 1728 documents (2683 virtual)
2018-10-03 08:48:59,239 : INFO : 28 batches submitted to accumulate stats from 1792 documents (2784 virtual)
2018-10-03 08:48:59,256 : INFO : 29 batches submitted to accumulate stats from 1856 documents (2838 virtual)
2018-10-03 08:48:59,268 : INFO : 30 batches submitted to accumulate stats from 1920 documents (2939 virtual)
2018-10-03 08:48:59,274 : INFO : 31 batches submitted to accumulate stats from 1984 documents (3014 virtual)
2018-10-03 08:48:59,286 : INFO : 32 batches submitted to accumulate stats from 2048 documents (3122 virtual)
2018-10-03 08:48:59,301 : INFO : 33 batches submitted to accumulate stats from 2112 documents (3223 virtual)
2018-10-03 08:48:59,310 : INFO : 34 batches submitted to accumulate stats from 2176 documents (3314 virtual)
2018-10-03 08:48:59,320 : INFO : 35 batches submitted to accumulate stats from 2240 documents (3338 virtual)
```

In [22]:
```
# define function to plot a graph to compare coherence scores
def evaluate_bar_graph(coherences, indices):
    """
    Function to plot bar graph.

    coherences: list of coherence values
    indices: Indices to be used to mark bars. Length of this and coherences should be equal.
    """
    assert len(coherences) == len(indices)
    n = len(coherences)
    x = np.arange(n)
    plt.bar(x, coherences, width=0.2, tick_label=indices, align='center', color=['red','red','red','green','green','green','blue'
    plt.xlabel('Models')
    plt.ylabel('Coherence Value')
```

In [23]:
```
# compare coherence scores
evaluate_bar_graph([lsi_no_pooling_coherence, lsi_user_pooling_coherence, lsi_hashtag_pooling_coherence,
                    hdp_no_pooling_coherence, hdp_user_pooling_coherence, hdp_hashtag_pooling_coherence,
                    lda_no_pooling_coherence, lda_user_pooling_coherence, lda_hashtag_pooling_coherence],
                   ['LSI1', 'LSI2', 'LSI3', 'HDP1', 'HDP2',
                    'HDP3', 'LDA1', 'LDA2', 'LDA3'])
plt.savefig("topic_model_comparison")
plt.show()
```



The HDP models have the best coherence scores but the topics are way too granular on inspection. In fact, the LDA achieves similar coherence scores if the number of topics is increased. But to ensure human interpretability, the number of topics is restricted to a lower number. In addition, the HDP models are very unstable on retraining. The hashtag pooling LDA model has by far the highest coherence score of the LSI and LDA models which is in line with the result after human inspection.

Jupyter Notebook 5: Apply best-performing LDA model

References

In [1]:
```
# https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
# https://radimrehurek.com/gensim/models/ldamodel.html
```

Prepare Notebook

In [2]:
```
# import packages
from gensim import models
import pandas as pd
import pickle
import logging
import os
from gensim import corpora, models
import numpy as np
import matplotlib.pyplot as plt
```

```
C:\Users\Sebastian Birk\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chunkize to c
hunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [3]:
```
# Log events
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

Load Test Documents

In [4]:
```
# load district documents for first objective
with open('tokenized_documents_district_pooling.p', 'rb') as fp:
    district_pooling_docs = pickle.load(fp)

# load month documents for second objective
# this one is not used
with open('tokenized_documents_month_pooling.p', 'rb') as fp:
    month_pooling_docs = pickle.load(fp)

# load month and district documents for second objective
with open('tokenized_documents_district_per_month_pooling.p', 'rb') as fp:
    district_per_month_pooling_docs = pickle.load(fp)
```

Load Model and Dictionary

In [6]:
```
# load model
lda_model =  models.LdaModel.load('lda_model_hashtag_pooling.model')

# load dictionary
if (os.path.exists("tourism_hashtag_pooling.dict")):
   dictionary = corpora.Dictionary.load('tourism_hashtag_pooling.dict')
   print("Hashtag pooling dictionary loaded!")
else:
   print("Please train LDA model first!")
```

```
2018-10-02 18:52:56,053 : INFO : loading LdaModel object from lda_model_hashtag_pooling.model
2018-10-02 18:52:56,058 : INFO : loading expElogbeta from lda_model_hashtag_pooling.model.expElogbeta.npy with mmap=None
2018-10-02 18:52:56,061 : INFO : setting ignored attribute state to None
2018-10-02 18:52:56,065 : INFO : setting ignored attribute dispatcher to None
2018-10-02 18:52:56,066 : INFO : setting ignored attribute id2word to None
2018-10-02 18:52:56,069 : INFO : loaded lda_model_hashtag_pooling.model
2018-10-02 18:52:56,071 : INFO : loading LdaState object from lda_model_hashtag_pooling.model.state
2018-10-02 18:52:56,076 : INFO : loaded lda_model_hashtag_pooling.model.state
2018-10-02 18:52:56,085 : INFO : loading Dictionary object from tourism_hashtag_pooling.dict
2018-10-02 18:52:56,094 : INFO : loaded tourism_hashtag_pooling.dict
Hashtag pooling dictionary loaded!
```

Running Model on Test Documents

In [7]:
```
# apply model to district pooled documents and get topic probability distributions
district_bow_list = [dictionary.doc2bow(text) for text in district_pooling_docs]
# district_1 = district_bow_list[0] # example code
district_topic_list = []

for index in range(len(district_bow_list)):
    district_bow = district_bow_list[index]
    topic_vector = lda_model[district_bow] # get topic probability distribution for a document
    district_topic_list.append(topic_vector)
```

In [8]:
```
# display district topic list
district_topic_list
```

Extract for Each Topic the Scores of the Districts

In [21]:
```
# initialize topic lists
topic0 = []
topic1 = []
topic2 = []
topic3 = []
topic4 = []
topic5 = []
topic6 = []
```

```
# save results in topic lists
for district in range(len(district_topic_list)):
    district_dict = dict(district_topic_list[district])

    if 0 in district_dict.keys():
        topic0.append(district_dict[0])
    else:
        topic0.append(0)

    if 1 in district_dict.keys():
        topic1.append(district_dict[1])
    else:
        topic1.append(0)

    if 2 in district_dict.keys():
        topic2.append(district_dict[2])
    else:
        topic2.append(0)

    if 3 in district_dict.keys():
        topic3.append(district_dict[3])
    else:
        topic3.append(0)

    if 4 in district_dict.keys():
        topic4.append(district_dict[4])
    else:
        topic4.append(0)

    if 5 in district_dict.keys():
        topic5.append(district_dict[5])
    else:
        topic5.append(0)

    if 6 in district_dict.keys():
        topic6.append(district_dict[6])
    else:
        topic6.append(0)
```

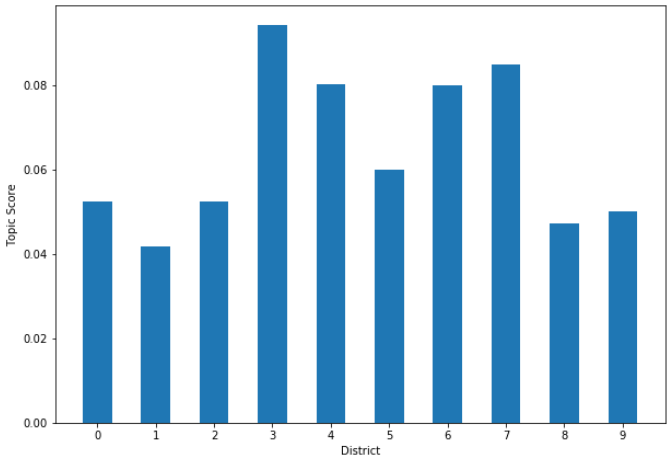
In [10]:
```
# check topic 0 to see whether the output looks as desired
topic0
```

Out[10]:
```
[0.052446194,
 0.041791383,
 0.052389883,
 0.094074719,
 0.08028692,
 0.059831791,
 0.079910465,
 0.084721953,
 0.047174171,
 0.050088074]
```

In [11]:
```
# define function to plot a graph to compare districts for a given topic
def district_bar_graph(districts, indices):
    """
    Function to plot district bar graph.

    districts: list with topic scores for each district
    indices: list of district number
    """
    assert len(districts) == len(indices)
    n = len(districts)
    x = np.arange(n)
    plt.figure(figsize=(10,7))
    plt.bar(x, districts, width=0.5, tick_label=indices, align='center')
    plt.xlabel('District')
    plt.ylabel('Topic Score')
```

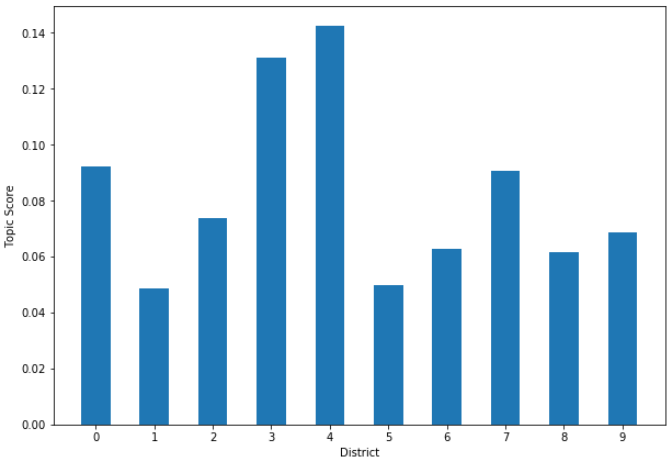
Topic 0: Sports, Health & Image

In [12]:
```
# plot topic scores
district_bar_graph(topic0,
                   list(range(len(topic0))))
plt.show()
```



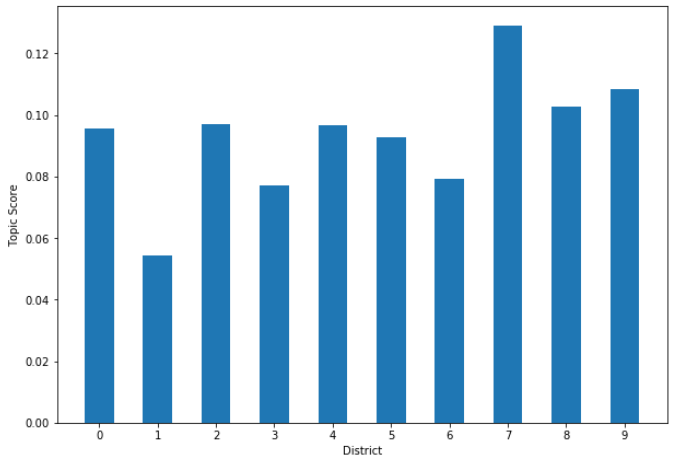
Topic 1: Lifestyle & Culture

In [13]:
```
# plot topic scores
district_bar_graph(topic1,
                   list(range(len(topic1))))
plt.show()
```



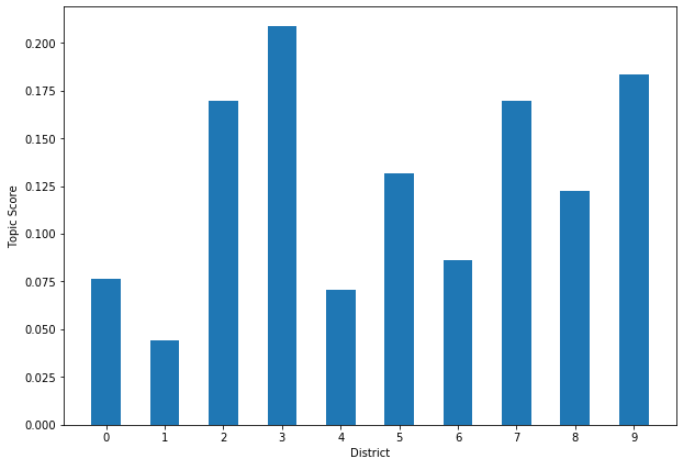
Topic 2: Nightlife

In [14]:
```
# plot topic scores
district_bar_graph(topic2,
                   list(range(len(topic2))))
plt.show()
```



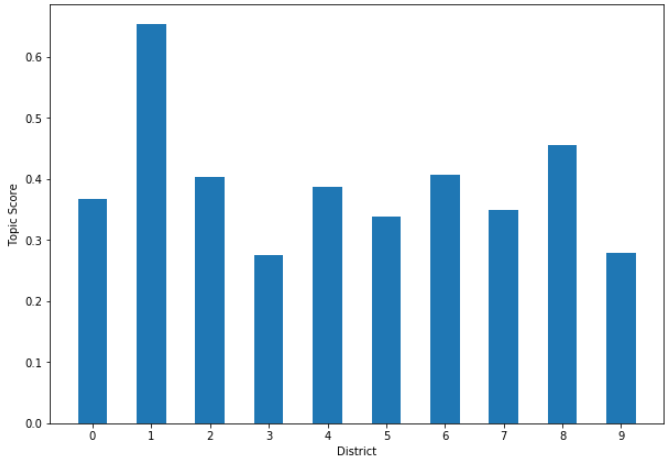
Topic 3: Streetart

In [15]:
```
# plot topic scores
district_bar_graph(topic3,
                   list(range(len(topic3))))
plt.show()
```



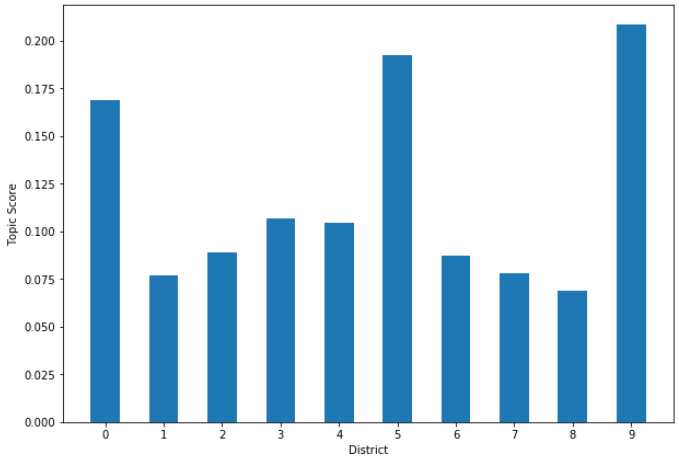
Topic 4: Sightseeing

In [16]:
```
# plot topic scores
district_bar_graph(topic4,
                   list(range(len(topic4))))
plt.show()
```



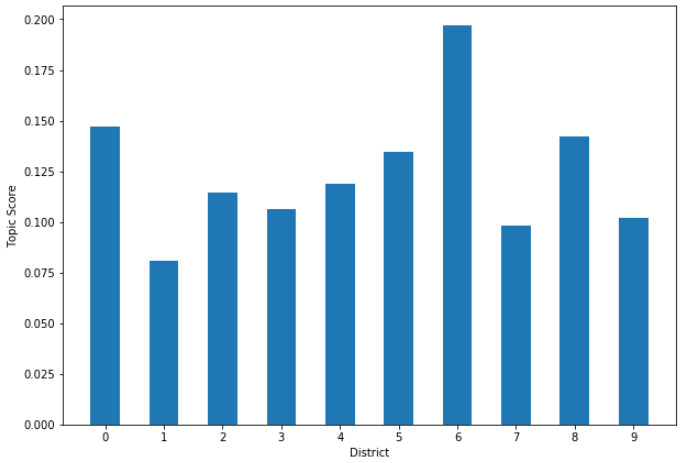
Topic 5: Summer, Sun & Friends

In [17]:
```
# plot topic scores
district_bar_graph(topic5,
                   list(range(len(topic5))))
plt.show()
```



Topic 6: Everyday Life

In [18]:
```
# plot topic scores
district_bar_graph(topic6,
                   list(range(len(topic6))))
plt.show()
```



Combine Results into a Dataframe

In [19]:
```
# create districts dataframe
dictionary_districts = {"district": ["01","02","03","04","05","06","07","08","09","10"],
                "topic0": topic0,
                "topic1": topic1,
                "topic2": topic2,
                "topic3": topic3,
                "topic4": topic4,
                "topic5": topic5,
                "topic6": topic6}
districts = pd.DataFrame(dictionary_districts)
```

In [22]:
```
# ignore this part

# display dataframe and set index
# districts.set_index("district")
```

Save Results to a CSV File

In [23]:
```
# ignore this part! (not necessary, is now done in the R script instead!)

# Ensure that excel will not delete leading 0s
# districts.district = districts.district.apply('"={}"'.format)
# districts.district = districts.district.apply(lambda x: str(x))
```

In [24]:
```
# display dataframe
districts
```

Out[24]:

	district	topic0	topic1	topic2	topic3	topic4	topic5	topic6
0	01	0.052446	0.092091	0.095601	0.076484	0.367206	0.168946	0.147225
1	02	0.041791	0.048469	0.054324	0.044437	0.653052	0.077020	0.080907
2	03	0.052390	0.073728	0.097064	0.169568	0.403647	0.089039	0.114564
3	04	0.094075	0.131077	0.076948	0.208561	0.276221	0.106959	0.106160
4	05	0.080287	0.142331	0.096636	0.070663	0.387017	0.104264	0.118803
5	06	0.059832	0.049774	0.092794	0.131546	0.339060	0.192691	0.134303
6	07	0.079910	0.062841	0.079290	0.086282	0.407509	0.087382	0.196785
7	08	0.084722	0.090782	0.128908	0.169783	0.349451	0.078087	0.098267
8	09	0.047174	0.061592	0.102585	0.122495	0.455181	0.069069	0.141904
9	10	0.050088	0.068616	0.108290	0.183360	0.279105	0.208409	0.102132

In [25]:
```
districts.to_csv("districts.csv", encoding='utf-8')
```

Next Part: Dynamic Analysis

In [26]:
```
# view district per month docs
district_per_month_pooling_docs
```

Out[26]:
```
[['boy',
  'hurt',
  'brain',
  'gothic',
  'quarter',
  'hearbeat',
  '#boyfriend',
  '#bf',
  '#love',
  '#partners',
  '#europe',
  '#catala',
  '#adeu',
  '#gordito',
  '#cosita',
  'sensibistro',
  '#tteeturns',
  '#sydorkosinspain',
  '#pinxtosforpip',
```

In [28]:
```
# delete last 2 docs (december)
del(district_per_month_pooling_docs[-1])
del(district_per_month_pooling_docs[-1])
len(district_per_month_pooling_docs)
```

Out[28]:
```
60
```

In [29]:
```
# define function to apply the model to unseen documents (similar procedure to above where it was applied to the district docs)
def run_lda_on_test_doc(docs):
    bow_list = [dictionary.doc2bow(text) for text in docs]
    topic_list = []

    for index in range(len(bow_list)):
        bow = bow_list[index]
        topic_vec = lda_model[bow]
        topic_list.append(topic_vec)

    return topic_list
```

In [30]:
```
# apply the model on dynamic data
# topics_month = run_lda_on_test_doc(month_pooling_docs)
topics_district_per_month = run_lda_on_test_doc(district_per_month_pooling_docs)
```

In [31]:
```
# display district per month topic list
topics_district_per_month
```

Out[31]:
```
[[(0, 0.044366464),
  (1, 0.061251059),
  (2, 0.099078745),
  (3, 0.068957314),
  (4, 0.36636451),
  (5, 0.25630867),
  (6, 0.1036732)],
 [(0, 0.041786052),
  (1, 0.053091906),
  (2, 0.076913379),
  (3, 0.056908067),
  (4, 0.63191766),
  (5, 0.074439704),
  (6, 0.064943202)],
 [(0, 0.037447397),
  (2, 0.20876357),
  (3, 0.23744458),
  (4, 0.21951687),
  (5, 0.1454431),
```

In [34]:
```
# initialize lists for topics
md_topic0 = []
md_topic1 = []
md_topic2 = []
md_topic3 = []
md_topic4 = []
md_topic5 = []
md_topic6 = []
```

```
# store results in topic lists
for month_district in range(len(topics_district_per_month)):
    district_month_dict = dict(topics_district_per_month[month_district])

    if 0 in district_month_dict.keys():
        md_topic0.append(district_month_dict[0])
    else:
        md_topic0.append(0)

    if 1 in district_month_dict.keys():
        md_topic1.append(district_month_dict[1])
    else:
        md_topic1.append(0)

    if 2 in district_month_dict.keys():
        md_topic2.append(district_month_dict[2])
    else:
        md_topic2.append(0)

    if 3 in district_month_dict.keys():
        md_topic3.append(district_month_dict[3])
    else:
        md_topic3.append(0)

    if 4 in district_month_dict.keys():
        md_topic4.append(district_month_dict[4])
    else:
        md_topic4.append(0)

    if 5 in district_month_dict.keys():
        md_topic5.append(district_month_dict[5])
    else:
        md_topic5.append(0)

    if 6 in district_month_dict.keys():
        md_topic6.append(district_month_dict[6])
    else:
        md_topic6.append(0)
```

In [35]:
```
# prepare lists with 'index'
district_list = ["01","02","03","04","05","06","07","08","09","10"] * 6
month_list = ["06"] * 10 + ["07"] * 10 + ["08"] * 10 + ["09"] * 10 + ["10"] * 10 + ["11"] * 10
```

In [36]:
```
# create districts per month dataframe
md_dictionary = {"district": district_list,
                 "month": month_list,
                 "topic0": md_topic0,
                 "topic1": md_topic1,
                 "topic2": md_topic2,
                 "topic3": md_topic3,
                 "topic4": md_topic4,
                 "topic5": md_topic5,
                 "topic6": md_topic6}
month_districts = pd.DataFrame(md_dictionary)
```

In [37]:
```
# display districts per month dataframe
month_districts
```

Out[37]:

	district	month	topic0	topic1	topic2	topic3	topic4	topic5	topic6
0	01	06	0.044366	0.061251	0.099079	0.068957	0.366365	0.256309	0.103673
1	02	06	0.041786	0.053092	0.076913	0.056908	0.631918	0.074440	0.064943
2	03	06	0.037447	0.000000	0.208764	0.237445	0.219517	0.145443	0.141546
3	04	06	0.020711	0.081662	0.056237	0.384384	0.066738	0.183288	0.206980
4	05	06	0.074042	0.076420	0.088062	0.076952	0.359566	0.177092	0.147865
5	06	06	0.047891	0.000000	0.105138	0.158698	0.369890	0.277315	0.031542
6	07	06	0.152659	0.040080	0.121856	0.000000	0.466459	0.014625	0.197907
7	08	06	0.211687	0.000000	0.000000	0.319650	0.392264	0.000000	0.073195
8	09	06	0.000000	0.000000	0.000000	0.000000	0.645816	0.169049	0.175504
9	10	06	0.040087	0.084793	0.124080	0.159790	0.259212	0.197681	0.134357
10	01	07	0.049183	0.111971	0.086070	0.058652	0.408366	0.134634	0.151124
11	02	07	0.033331	0.048663	0.058142	0.041569	0.664082	0.069131	0.085082
12	03	07	0.062612	0.065308	0.123157	0.231916	0.312460	0.107327	0.097220
13	04	07	0.089442	0.070237	0.026378	0.317535	0.325589	0.090152	0.080668
14	05	07	0.135451	0.317992	0.060961	0.145157	0.179542	0.054685	0.106213

Save Results to a CSV File

In [38]:
```
month_districts.to_csv("month_districts.csv", encoding='utf-8')
```

In [39]:
```
# verify results
topics_district_per_month[10] # month 7, district 1
```

Out[39]:
```
[(0, 0.049182747),
 (1, 0.11197082),
 (2, 0.086070046),
 (3, 0.058652479),
 (4, 0.40836591),
 (5, 0.13463441),
 (6, 0.15112357)]
```

In [40]:
```
# verify results
district_per_month_pooling_docs[10] # month 7, district 1
```

Out[40]:
```
['#beautifulmemories',
 '#artdirector',
 '#lovetravel',
 '#inspiration',
 '#beautifulcity',
 'free',
 'water',
 '#goticquarterbarcelona',
 '#photography',
 '#traveltheworld',
 'gothic',
 'quarter',
 'new',
 'friend',
 "can't",
 'enough',
 'tapa',
 'sangria',
 'maryland',
```

R Script 1: Create maps of Barcelona districts

bcn_district_mapping.R

```r
## install packages (skip this part once installed!)
install.packages("sp")
install.packages("spdep")
install.packages("rgdal")
install.packages("units", type='binary')
install.packages("gdalUtils")
install.packages("tmap")

## import packages
library(sp)
library(spdep)
library(rgdal)
library(tmap)
library(RColorBrewer)
library(dplyr)
library(ggplot2)
library(data.table)
library(stringr)
```

```r
## load districts, neighbourhoods and tweets data
file1 = "bcn_districts.RData"
load(file1, envir = parent.frame(), verbose = FALSE)

file2 = "tweets.RData"
load(file2)

file3 = "bcn_neighbourhoods.RData"
load(file3)

## save data in dataframes to view and inspect the data
df_districts <- bcn.di@data
df_neighbourhoods <- bcn.nb@data

# filter tweets by English language
english_tweets <- subset(tweets, language3=="ENGLISH")

## count tweets per district
tweets_count <- english_tweets[, .(count = .N), by = .(id_distrito)]
# remove first characters of district column
tweets_count$id_distrito <- str_sub(tweets_count$id_distrito,-2,-1)
```

```r
# join district map and tweet count data on district level for plot
districts <- bcn.di
districts@data <- merge(districts@data, tweets_count, by.x="C_Distri", by.y="id_distrito")
# check results
districts@data
# remove special characters from district name
districts@data$N_Distri <- c("Ciutat Vella", "Eixample", "Sants-Montjuic", "Les Corts", "Sarria-Sant Gervasi", "Gracia", "Horta-
# create plot
qtm(districts, fill=colnames(districts@data)[12], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), titl
# save plot as file
dev.print(jpeg, filename="number_of_tweets_per_district.jpg", width=1000)

## plot map of Barcelona districts
# create plot
qtm(districts, fill=colnames(districts@data)[2], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="Spectral"), tit
# save plot as file
dev.print(jpeg, filename="districts_of_bcn.jpg", width=1000)
```

```r
# load csv file that contains the results of the topic analysis
results = read.csv("districts.csv", header = TRUE)
# inspect the dataframe
results
# display mean topic scores
mean(results$topic0)
mean(results$topic1)
mean(results$topic2)
mean(results$topic3)
mean(results$topic4)
mean(results$topic5)
mean(results$topic6)
# add leading zeros for merge
results$district = list("01","02","03","04","05","06","07","08","09","10")
```

```r
# join district map and topic analysis results
districts@data <- merge(districts@data, results, by.x="C_Distri", by.y="district")
# check dataframe
districts@data
```

```
79
80 # plot results for topic0
81 qtm(districts, fill=colnames(districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
82 # save plot as file
83 dev.print(jpeg, filename="topic0_per_district.jpg", width=1000)
84
85 # plot results for topic1
86 qtm(districts, fill=colnames(districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
87 # save plot as file
88 dev.print(jpeg, filename="topic1_per_district.jpg", width=1000)
89
90 # plot results for topic2
91 qtm(districts, fill=colnames(districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
92 # save plot as file
93 dev.print(jpeg, filename="topic2_per_district.jpg", width=1000)
94
95 # plot results for topic3
96 qtm(districts, fill=colnames(districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
97 # save plot as file
98 dev.print(jpeg, filename="topic3_per_district.jpg", width=1000)
99
```

```
99
100 # plot results for topic4
101 qtm(districts, fill=colnames(districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
102 # save plot as file
103 dev.print(jpeg, filename="topic4_per_district.jpg", width=1000)
104
105 # plot results for topic5
106 qtm(districts, fill=colnames(districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
107 # save plot as file
108 dev.print(jpeg, filename="topic5_per_district.jpg", width=1000)
109
110 # plot results for topic6
111 qtm(districts, fill=colnames(districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), title
112 # save plot as file
113 dev.print(jpeg, filename="topic6_per_district.jpg", width=1000)
114
```

R Script 2: Create dynamic maps of Barcelona districts

bcn_district_mapping_dynamic.R

```r
## import packages
library(sp)
library(spdep)
library(rgdal)
library(tmap)
library(RColorBrewer)
library(dplyr)
library(ggplot2)
library(data.table)
library(stringr)
library(grid)

## load districts, neighbourhoods and tweets data
file1 = "bcn_districts.RData"
load(file1, envir = parent.frame(), verbose = FALSE)

file2 = "tweets.RData"
load(file2)

file3 = "bcn_neighbourhoods.RData"
load(file3)
```

```r

## save data in dataframes to view and inspect the data
df_districts <- bcn.di@data
df_neighbourhoods <- bcn.nb@data

# load csv file that contains the results of the dynamic topic analysis
res = read.csv("month_districts.csv", header = TRUE)
# inspect the dataframe
res
# add leading zeros for merge
res$district = rep(list("01","02","03","04","05","06","07","08","09","10"),6)

# find max values for each topic
max0 <- max(res$topic0)
max1 <- max(res$topic1)
max2 <- max(res$topic2)
max3 <- max(res$topic3)
max4 <- max(res$topic4)
max5 <- max(res$topic5)
max6 <- max(res$topic6)

# extract months
jun <- res[1:10,]
jul <- res[11:20,]
aug <- res[21:30,]
sep <- res[31:40,]
oct <- res[41:50,]
nov <- res[51:60,]
```

```r

# join district map and topic analysis results for june
june_districts <- bcn.di
june_districts@data <- merge(june_districts@data, jun, by.x="C_Distri", by.y="district")
# check dataframe
june_districts@data

# join district map and topic analysis results for july
july_districts <- bcn.di
july_districts@data <- merge(july_districts@data, jul, by.x="C_Distri", by.y="district")
# check dataframe
july_districts@data

# join district map and topic analysis results for august
august_districts <- bcn.di
august_districts@data <- merge(august_districts@data, aug, by.x="C_Distri", by.y="district")
# check dataframe
august_districts@data

# join district map and topic analysis results for september
september_districts <- bcn.di
september_districts@data <- merge(september_districts@data, sep, by.x="C_Distri", by.y="district")
# check dataframe
september_districts@data
```

```r
# join district map and topic analysis results for october
october_districts <- bcn.di
october_districts@data <- merge(october_districts@data, oct, by.x="C_Distri", by.y="district")
# check dataframe
october_districts@data

# join district map and topic analysis results for november
november_districts <- bcn.di
november_districts@data <- merge(november_districts@data, nov, by.x="C_Distri", by.y="district")
# check dataframe
november_districts@data
```

```r
## plot results for topic 0
current_topic_max = max0
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_max

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[14], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
print(plot0, vp = vplayout(1, 1))
```

```r
# save plot as file
dev.print(jpeg, filename="topic0_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal(n=10
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[14], borders='black', fill.n=10, fill.palette=brewer.pal
```

```r
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic0_per_month_per_district.jpg", width=1000)

## plot results for topic 1
current_topic_max = max0
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_max

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[15], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))
```

```r
# save plot as file
dev.print(jpeg, filename="topic1_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal(n=10
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[15], borders='black', fill.n=10, fill.palette=brewer.pal

# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))
```

```r
# save plot as file
dev.print(jpeg, filename="topic1_per_month_per_district.jpg", width=1000)

## plot results for topic 2
current_topic_max = max2
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_ma

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[16], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))

# save plot as file
dev.print(jpeg, filename="topic2_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal(n=1
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[16], borders='black', fill.n=10, fill.palette=brewer.pal
```

```r
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic2_per_month_per_district.jpg", width=1000)
```

```r
## plot results for topic 3
current_topic_max = max3
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_ma

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[17], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))

# save plot as file
dev.print(jpeg, filename="topic3_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal(n=1
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[17], borders='black', fill.n=10, fill.palette=brewer.pal
```

```r
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic3_per_month_per_district.jpg", width=1000)
```

```r
## plot results for topic 4
current_topic_max = max4
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_ma

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[18], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))

# save plot as file
dev.print(jpeg, filename="topic4_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal(n=1
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[18], borders='black', fill.n=10, fill.palette=brewer.pal
```

```
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic4_per_month_per_district.jpg", width=1000)
```

```
## plot results for topic 5
current_topic_max = max5
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_ma

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[19], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))

# save plot as file
dev.print(jpeg, filename="topic5_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal(n=1
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[19], borders='black', fill.n=10, fill.palette=brewer.pal
```

```
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic5_per_month_per_district.jpg", width=1000)
```

```
## plot results for topic 6
current_topic_max = max6
manual_scale = c(0, current_topic_max*0.1, current_topic_max*0.2, current_topic_max*0.3, current_topic_max*0.4, current_topic_ma

plot0 <- qtm(june_districts, fill=colnames(june_districts@data)[20], borders='black',
             fill.n=10, fill.palette=brewer.pal(n=10, name="YlOrRd"), scale=2, title="June", fill.style="fixed", fill.breaks=manu

grid.newpage()
print(plot0, vp = vplayout(1, 1))

# save plot as file
dev.print(jpeg, filename="topic6_per_month_per_district_legend.jpg", width=1000)

plot1 <- qtm(june_districts, fill=colnames(june_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot2 <- qtm(july_districts, fill=colnames(july_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal(n=10, na
plot3 <- qtm(august_districts, fill=colnames(august_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal(n=1
plot4 <- qtm(september_districts, fill=colnames(september_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pa
plot5 <- qtm(october_districts, fill=colnames(october_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal(n=
plot6 <- qtm(november_districts, fill=colnames(november_districts@data)[20], borders='black', fill.n=10, fill.palette=brewer.pal
```

```
# prepare multiple plots
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,3)))
print(plot1, vp = vplayout(1, 1))
print(plot2, vp = vplayout(1, 2))
print(plot3, vp = vplayout(1, 3))
print(plot4, vp = vplayout(2, 1))
print(plot5, vp = vplayout(2, 2))
print(plot6, vp = vplayout(2, 3))

# save plot as file
dev.print(jpeg, filename="topic6_per_month_per_district.jpg", width=1000)
```

More Information on Topic Modeling

Labeling topics

It is important to specify that topics are not automatically labeled in common topic modeling approaches, as they are extracted with an unsupervised technique that is not aware of the meaning of the topics. Several works have tried to label topics (Magatti et al., 2009; Lau et al., 2011; Aletras and Stevenson, 2014). However, most of the benefits of topic models can be exploited even if the meaning of the topics is not known or if they are labeled manually.

pLSI Topic Model

In order to account for the deficits of the LSI model, the next step in the timeline of topic modeling was an extension of the LSI model by the so called probabilistic Latent Semantic Indexing (pLSI) model (Hofmann, 1999). PLSI actually has little to do with LSI other than the fact that they perform roughly the same objective and can be used in similar applications (Gimpel, 2006). In contrast to standard LSI, which uses SVD, the probabilistic variant of LSI has a solid statistical foundation and defines a proper generative data model, i.e. a procedure for generating documents using a series of probabilistic steps (Hofmann, 2017)[22]. The core of pLSA is a statistical mixture model which has been called aspect model and which tries to extract the latent components/ topics by reversing this generative data process through statistical inference. Given a collection of documents,

[22] The generative model can be defined in the following way:
- Select a document with probability P(d)
- Pick a latent class z with probability P(z|d)
- Generate a word w with probability P(w|z)

(Hofmann, 2017).

probability distributions over the topics can be obtained and the model can be used for topic detection in new documents (Gimpel, 2006). Retrieval experiments on a number of test collections indicate substantial performance gains of pLSI over standard LSI. This can be contributed to the fact that pLSI is based on the likelihood principle, defines a generative data model and thus directly minimizes word perplexity (Hofmann, 2017). One major drawback of the pLSI model, however, is that the generative model is defined in a way so that possible topic proportions for a document are learned from the document collection (the text corpus) itself as opposed to allowing for more flexible arbitrary topic proportions from a prior probability distribution. While this is not a problem for the original purpose of the pLSI model of text retrieval[23], in more advanced applications, such as text classification, it is crucial to have a model flexible enough to properly handle text that has not been seen before, i.e. text outside of the training set (Gimpel, 2006; Blei et al., 2003). In addition, as a topic distribution must be learned for each document in the collection, the number of parameters of the pLSI model grows with the number of documents, basically rendering the model useless for an application using social media data where document collection sizes can be on the order of a billion documents.

Hierarchical topic models

Variations of LDA can be found in hierarchical topic models, which can be used as an extension to LDA models to account for the fact that standard LDA models can sometimes be too rigid on their own to reflect the open-ended nature of datasets which often grow over time and as they grow bring new entities and structures to the fore. In general, hierarchical topic models specify a generative probabilistic model for hierarchical

[23] In information retrieval the current document collection to be stored can be viewed as a fixed collection.

structures and take a Bayesian perspective on the problem of learning these structures from the data by using an algorithm that constructs the hierarchy as data are made available. They use a structured prior on the topics underlying a corpus of documents, with the aim of bringing more order to an unstructured set of thematic concepts (Paisley et al., 2014). One of the first models following this paradigm was the hierarchical LDA (hLDA) proposed by Blei et al. in 2003. HLDA is, as the name suggests, an adaptation of the LDA technique that models topics as mixtures of a new, distinct level of topics, drawn from dirichlet distributions (and not processes). Similar to LDA, it still treats the number of topics as a hyperparameter, i.e., independent of the data. The difference is that the clustering is now hierarchical: it learns a clustering of the first set of topics themselves, giving more general, abstract relationships between topics (and hence, words and documents). For example, it can be thought of as clustering the stack exchanges into “math”, “science”, “programming”, “history”, etc. as opposed to clustering “data science” and “cross validation” into an abstract “statistics and programming” topic that shares some concepts with, say, “software engineering”, but the “software engineering” exchange is clustered on a more concrete level with the “computer science” exchange, and the similarity between all of the mentioned exchanges does not appear as much until the upper layer of clusters. Future research could look into the potential of the hLDA model for social media and specifically Twitter data.